CFPOST User's Guide[*]

The NPARC Alliance

NASA Glenn Research Center
Cleveland, Ohio

USAF Arnold Engineering Development Center
Tullahoma, Tennessee

[*]This is an unnumbered version of this document, created June 2, 2016. It is essentially identical in content to the 1 Feb 1998 version of the "CFPOST Reference Manual," by the Boeing Company. Please send corrections, additions, ideas, etc., to Mark Fisher at *mark.s.fisher@boeing.com* or Dennis Yoder at *Dennis.A.Yoder@nasa.gov*.

Contents

1 General Overview

1.1 Introduction

CFPOST is a post-processing tool for analyzing CFD results. It reads as input a Common Grid (*.cgd*) and Common Flow (*.cfl*) file, and may be used to perform many varied functions, such as listing and plotting results, generating reports, and producing files for other plotting packages and post-processors. Hardcopy plot output is available in Boeing intermediate plot file (IPF) or PostScript formats.

CFPOST provides commands for the user to precisely specify the information of interest (the `variable` command), the domain of interest (the `zone`, `subset`, `surface`, `cut`, and `rake` commands), and the units of measure in which the results are to be presented (the `units` and `variable` commands).

1.2 Command Syntax

All CFPOST commands are of the form:

```
command keyword [value] keyword [value] ...
```

In this guide all commands and keywords will appear in a `fixed-pitch` font and user specified values will appear in an *italic* font. Commands and keywords can be abbreviated but such abbreviations should be limited to three characters. Words like `and`, `at`, and `to` cannot be abbreviated. Lists enclosed in '`{`' and '`}`' indicate that exactly one item from the list must be selected. Items enclosed in '`[`' and '`]`' indicate optional items. If a list is enclosed in '`[`' and '`]`' then only one of the list entries may be optionally selected.

Commands may be continued by terminating the line to be continued with a '`-`' (hyphen) and continuing on the next line. This may be repeated for as many lines as necessary. For example:

```
integrate force -
   output force.lis -
   reference moment 100.0 0.0 10.3 -
   reference area 95.0
```

Comments may also be included. On any command, all characters following a '`!`' are ignored. For example:

```
! This is a standalone comment statement
subset j all k all i 1    ! This is a trailing comment
```

Embedded tabs and blanks are ignored as are completely blank lines.

For the most part CFPOST is case insensitive, i.e., upper and lower case characters may be freely intermixed. There are two exceptions however. Filenames must be entered in the correct case on systems where filenames are case sensitive (i.e., Unix). Note that if a suffix (*.lis*, *.com*, etc.) is not provided on a filename, CFPOST will automatically provide one and it will be lower case. Thus, if a file has an upper case suffix it must be specified explicitly. The second exception where case is important is variable names on the `variable` command.

1.3 Processing Overview

A CFPOST session is generally divided into a well-ordered sequence of commands as follows:

- `grid` and `solution` commands to define the input files to be processed.
- A `units` command to define the default system of output units.
- `zone`, `subset`, `cut`, and `rake` commands to define the domain of the input files to be processed.
- A `variable` command to define the information requested.
- A processing command to generate the requested information.

The commands that comprise the above sequence will be more fully described in the following sections.

2 General Purpose Commands

`@` — Invoke a file containing CFPOST commands

`@`*filename*

filename The name of a file that contains CFPOST commands. All of the commands within the file are executed and then processing resumes with the next command after the `@` command. A file that is invoked by an `@` command may itself invoke other files with `@` commands, but may not invoke any file that is currently being processed by a higher `@` command.

`boundary layer` — Specify method to determine the boundary layer edge

```
boundary layer edge is percent percent of {uinf|umax|p0inf|h0inf} |
boundary layer edge is percent percent of -
   {deltau|deltaq|deltak|deltauv|deltauw|deltavw} |
boundary layer edge is {p0|T0|y+} of value
```

The `boundary layer` command is used to specify how the location of the boundary layer edge is determined when computing the variable `delta`. Note this is not used for the computation of any other boundary layer thickness variables. The default specification is 99% of `umax`.

There are three possible methods for determining the boundary layer edge, as follows:

```
boundary layer edge is percent percent of {uinf|umax|p0inf|h0inf}
```

With this method, the boundary layer edge is located where the velocity magnitude (or total pressure for `p0inf`, or total enthalpy for `h0inf`) first becomes greater than or equal to a percentage of a specified reference value. `umax` signifies the maximum velocity magnitude on the grid line normal to the surface.

```
boundary layer edge is percent percent of -
   {deltau|deltaq|deltak|deltauv|deltauw|deltavw}
```

With this method, the boundary layer edge is located where the percentage difference in velocity (or dynamic pressure, or k of k-ϵ, or cross flow velocity) magnitude between two adjacent points on the same normal grid line first becomes less than the specified value.

```
boundary layer edge is {p0|T0|y+} of value
```

With this method, the boundary layer edge is located where the total pressure, total temperature, or y^+ first becomes greater than or equal to the specified value. *value* must be specified in the currently selected units.

Examples

```
     boundary layer edge is 99 percent of uinf
     boundary layer edge is 1 percent of deltau
     units english inches
     boundary layer edge is p0 of 14.6    ! 14.6 psi
```

`clear` — Clear selections

```
clear [all] [cuts] [rakes] [subsets] [variables] [chemistry]
```

The `clear` command removes all selections from the specified lists. The result is that the specified lists are initialized to the same state as they were at the time CFPOST was started.

Note that currently `clear all` does not clear the chemistry data.

See Also: `cut`, `rake`, `subset`, `surface`, `group`, `variable`, `chemistry` `exit, quit` — Exit from CFPOST

```
exit | quit
```

The `exit` or `quit` command results in the termination of this CFPOST session. Control is returned to the operating system.

`first order, second order` — Set order for differentiation

```
first order | second order
```

The `first order` and `second order` commands establish the order to be used when differentiating variables in the derivation of `Cf`, `Cfx`, `Cfy`, `Cfz`, `tau`, `taux`, `tauy`, `tauz`, and `Q`. The default value is `second order`.

`orientation` — Specify axes orientation

```
orientation [up [axis] {x|y|z}] [downstream [axis] {x|y|z}]
```

`up [axis] {x|y|z}`	Specifies which of the three coordinate directions represents the 'up' direction.
`downstream [axis] {x|y|z}`	Specifies which of the three coordinate directions represents the 'downstream' direction.

The `orientation` command is used to specify which coordinate direction is up and which is downstream. This information is necessary for computing lift and drag in the `integrate force` command and for computing the `localpha` and `locbeta` variables when requested. If the `orientation` command is entered without parameters, the current selections are displayed.

See Also: `integrate force`

`show` — Show current selections

```
show [all] [cuts] [rakes] [subsets] [variables] [chemistry]
```

The `show` command shows the current variables, chemistry data, subsets, and selections as well as the currently enabled units of measure.

See Also: `rake`, `subset`, `surface`, `variable`, `chemistry` `spawn, system` — Invoke a subprocess

```
[spawn [command] | system [command]]
```

The `system` and `spawn` commands allow a user to temporarily suspend operation of CFPOST and invoke one or more operating system commands. If *command* is specified, the command is executed and control is returned to CFPOST. If *command* is not specified, the user is left at the operating system prompt and may issue as many operating system commands as desired. Control is returned to CFPOST when the system dependent sub-process termination command is issued (typically `exit` or `logoff`).

`type` — Display the contents of a text file

```
type filename [lines lines-per-page] [pause | nopause]
```

filename	The name of the file to be displayed.
`lines` *lines-per-page*	Specifies the number of lines of the listing to be displayed on each screen. This value should be one less than the number of lines on the screen. The default value is 23.
`pause | nopause`	Specifies whether the program should pause at the end of each screen full of data. The default is to pause at the end of each screen.

The `type` command displays the contents of the specified file on the screen. This command is useful for viewing the output file from a previous command such as `integrate` or `list`. The default extension is *.lis* if not specified.

3 File Specification Commands

`grid` — Specify the name of the grid (*.cgd*) file

```
grid file [mode {read_only|read_write}]
```

file The name of the file to be used for retrieving geometry data during subsequent processing. If a `grid` command is not specified then any reference to the geometry data will be resolved from the file specified on the `solution` command.

The `mode` option may be used to specify that the file is to be opened in either read-only or read-write mode. The default is read-only mode.

The `mode` option is only applicable in CFPOST versions 4.1 and later. Earlier versions automatically open the file in read-write mode, and a file read error will occur if the `mode` option is used.

Examples

```
grid test.cgd
grid /usr/joe/abc.cgd
```

`solution` — Specify the name of the solution (*.cfl*) file

```
solution file [mode {read_only|read_write}]
```

file The name of the file to be used for retrieving non-geometry data during subsequent processing. Geometry data will also be retrieved from this file if a `grid` command has not been specified.

The `mode` option may be used to specify that the file is to be opened in either read-only or read-write mode. The default is read-only mode. Certain CFPOST commands (e.g., `calculate`), require that the solution file be opened in read-write mode.

The `mode` option is only applicable in CFPOST versions 4.1 and later. Earlier versions automatically open the file in read-write mode, and a file read error will occur if the `mode` option is used.

Examples

```
solution test.cfl
solution /usr/joe/abc.cfl
```

`chemistry` — Specify the name of the chemistry data (*.chm*) file

```
chemistry file file [itest number mode [number mode [...]]]
```

file The name of the file to be used for retrieving chemistry data during subsequent processing.

This file should be the same *.chm* file used during the flow solver run. It contains species data for thermodynamic properties, reaction rates, and transport properties, and is required if the solution (*.cfl*) file is from a Wind-US run with frozen or finite-rate chemistry.

The `itest` option may be used to specify chemistry-related `TEST` options that were used during the Wind-US run. Currently the only supported option number in CFPOST is 71, which controls how thermodynamic and transport properties are calculated when the temperature is outside the range of the curve fit equations. See the "Test Options" section in the *Wind-US User's Guide* for more details.

Examples

```
chemistry file test.chm
chemistry file /usr/joe/abc.chm
```

See Also: The `show chemistry` and `clear chemistry` commands for displaying the species information and clearing that data when no longer needed.

4 Domain Selection Commands

`zone` — Specify zones to be processed

```
zone {zone|zone range} [limit xmin xmax ymin ymax zmin zmax]
```

zone Specifies the zone (block) to which subsequent `subset` and `surface` commands apply. All `subset` and `surface` commands following a `zone` command up to the next `zone` command specify the parts of the zone to be included in subsequent processing. If no `subset` or `surface` commands appear, then the default subset(s) will be used. The default subsets are those defined by `subset` and `surface` commands that occur before the first `zone` command.

zone range Specifies the zone range to which only the default subsets apply. Any subset commands specified after the zone range command apply only to the last zone! The range format is *n* [*to*|*thru*|*through*] *m* where *n* is a zone number and *m* is a zone number or the keyword `last`. Note that to use the `last` keyword a grid or flow file must be opened and that the zone list does not change when a new file is opened.

`limit` *xmin xmax ymin ymax zmin zmax*
Specifies the zone list generated is restricted to zones that fall within the specified min-max box. The units of *xmin*, *xmax*, etc., are in the length units that was in effect at the time of the command. Note that a grid file must be opened and that the zone list does not change when a new file is opened.

Example 1

```
zone 1                      ! Select zone 1
   subset k all i all j 1   ! Applies to zone 1
   subset k all i all j last ! Applies to zone 1
zone 3                      ! Select zone 3
   subset j all k all i 1   ! Applies to zone 3
list                        ! Do a command on the above
```

Example 2

```
subset k all i all j 1    ! Create a default subset
zone 1                    ! Select zone 1, j = 1
zone 3                    ! Select zone 3, i = 1
   subset j all k all i 1 ! Applies to zone 3
zone 4 to last            ! Select zones 4 to last, j = 1 for all zones
artis output test iris4   ! Initiate processing
```

In Example 2 a default subset is defined (the $j = 1$ surface). Since no `subset` commands are included in the scope of zones 1 and 4, the default subset will be used. The default subset will *not* be used in zone 3 since a `subset` command was specified in its scope.

Example 3

```
unit inches      ! Set current length units to inches
zone 1 to last limit -1.0 1.0 -5.0 5.0 1.0 2.0
                 ! Set range to all zones, but only process those within specified limits
```

See Also: The `subset` and `surface` commands for how to define subsets; the `clear` command for how to clear the subset list; the `show` command to determine the current units.

`subset` — Specify a subset of a zone to be processed

```
subset range1 range2 range3
```

range Specifies a coordinate direction and indices to be included in the subset. The format of *range* is:

`{i|j|k|u}` *indices* `[;` *indices* `...]`

The coordinate direction or subset type (`i`, `j`, `k` for structured, `u` for unstructured) is selected by specifying `i`, `j`, `k` or `u`. *indices* specifies the indices and is of the form:

`start [,`*end* `[,`*incr*`]]`

`last` signifies the maximum value of the specified coordinate direction in the zone in which the subset appears and may be used in place of *end*. `all` is an alternative form of *1,last* and may be used in place of the *start* and *end* values. *incr* is assumed to be *1* if not specified.

Only one range specification is allowed for unstructured grids.

The `subset` command specifies a portion of the input files to be included in subsequent processing. All three coordinate directions must appear exactly once, one in each range. The order of appearance of the coordinate directions in the range's is important as we will see shortly.

Some CFPOST commands process lines of data (e.g.: `genplot`), some expect surfaces (e.g.: `artis`, `cfdfem`, `integrate`) and some expect volumes (e.g.: `plot3d`). Since we are dealing with three-dimensional data, we must have a way to signify what coordinate direction is defining a desired line or what two coordinate directions define a surface. Furthermore, some functions require the definition of a normal vector.

In FORTRAN terminology we would define a line or a curve with:

```
DIMENSION A(IDIM)
```

A surface would be defined as:

```
DIMENSION A(IDIM,JDIM)
```

And a volume would be defined as:

```
DIMENSION A(IDIM,JDIM,KDIM)
```

That is, a surface can be considered as consisting of `JDIM` lines of `IDIM` points. Furthermore, a volume is `KDIM` surfaces consisting of `JDIM` lines of `IDIM` points.

Continuing the FORTRAN analogy, the components of *range1* become `IDIM`, *range2* become `JDIM`, and *range3* become `KDIM`. Thus, for functions that require a curve, *range1* defines the varying coordinate direction. For functions that require a surface *range1* and *range2* define the two varying coordinate directions.

Some functions, and the computation of some variables, require the knowledge of a normal direction. The normal direction coordinate is the coordinate specified in *range3*. The actual direction of the normal vector is computed by taking the cross product $\mathbf{A} \times \mathbf{B}$ where $\mathbf{A}$ is a vector increasing towards `IDIM` and $\mathbf{B}$ is a vector increasing towards `JDIM`.

Note: For 2-D solutions (KDIM = 1), you must always specify the `k` component of the subset to be '`k` *all*' rather than '`k` *1*'.

Example 1 - Defining lines

```
subset i 1,10 k 1 j 10     ! The first index must vary
```

For unstructured grids the indicies specify a range of points in the variable arrays to be considered as a one dimensional structured array.

Example 2 - Defining surfaces

```
subset k all i all j 1       ! Surface j = 1
subset i all k all j last    ! Surface j = JMAX
```

Note that in these two cases, the order of specifying the first two coordinate directions was chosen so that the normal vector would be pointing into the control volume.

Unstructured surfaces must be defined using the `surface` command because there is no contiguous range of points which define a surface.

For a stack of surfaces, you simply enter:

```
subset j all k all i 1,11,2     ! A group of planes
```

Example 3 - Defining volumes

```
subset i all,2 j 1,11,2;15,last k 1;11;18;27
```

In the preceding example, every other point in the i direction is included. In the j direction every other point in the range 1 to 11 and every point from 15 to the end is included. In the k direction only points 1, 11, 18, and 27 are included.

See Also: The `surface` command for a simpler method to define subsets that are surfaces; the `clear` command for how to clear the subset list; the `zone` command for how to select the current zone.

`surface` — A simple way to define a surface subset

```
surface { {i|j|k} {1|last} [range1 [range2]] | u surface_number }
```

`{i|j|k}` `{`*1*`|`*last*`}`	Specifies the computational coordinate and index of the surface to be defined.
`[`*range1* `[`*range2*`]]`	Optionally specifies the computational coordinates and indices of the portion of the surface to be included. *range* is defined exactly like *range* in the `subset` command except that the indices must be specified so that they are monotonically increasing. If a computational coordinate range is omitted then all indices for that coordinate direction are included.
`u` *surface_number*	Specifies the identifying number of the desired unstructured surface.

The `surface` command is a simpler method for defining surface subsets for those commands that require surfaces as input. The advantage of this command over the `subset` command is that it will correctly order the coordinate indices so that the normal vector will automatically be directed into the volume.

For unstructured grids, a surface is identified by an integer surface identification number. The unstructured surface is defined by a range of cell faces. The points that define these faces are automatically extracted from the point list.

Examples

`surface i` *1*	is the same as	`subset j` *all* `k` *all* `i` *1*
`surface i` *last*	is the same as	`subset k` *all* `j` *all* `i` *last*
`surface j` *1*	is the same as	`subset k` *all* `i` *all* `j` *1*
`surface j` *last*	is the same as	`subset i` *all* `k` *all* `j` *last*
`surface k` *1*	is the same as	`subset i` *all* `j` *all* `k` *1*
`surface k` *last*	is the same as	`subset j` *all* `i` *all* `k` *last*
`surface k` *1* `j` *1,10*	is the same as	`subset i` *all* `j` *1,10* `k` *1*

See Also: The `subset` command for a more complete definition of ranges and normal vectors.

`group` — Use a named surface group

```
group groupname
```

Surface groups can be defined with the *cfpart* utility and stored in the grid (*.cgd*) file. The `group` command can be used to reference the surface information stored under *groupname*. The primary advantage of using a surface group is to define a physical entity (i.e., wing) which is composed of multiple computational surfaces and may span multiple zones.

`cut` — Specify a cutting plane

```
cut [at] {x|y|z} xyz |
cut [at] x1 y1 z1 x2 y2 z2 x3 y3 z3 |
cut [at] point px py pz vector vx vy vz
```

The `cut` command defines a cutting plane that will be passed through all of the active subsets. The intersection points will be passed on to the command processor as the data to be processed. The `cut` command can only be used with certain commands such as `genplot`, `genplot surface`, `integrate area` and `integrate flux`. If the intersection is a plane, the resultant polygons will have their points ordered so that the points will be traversed in a counter-clockwise directions if the polygon is viewed from the side of the plane toward which the normal vector points.

Up to 192 cut/crinkle planes may be defined. The list of cut and crinkle planes can be cleared with the `clear` command.

There are three possible formats for the `cut` command, as follows:

```
cut [at] {x|y|z} xyz
```

`{x|y|z}` *xyz* Specifies the location of a plane that is normal to the specified coordinate axis. The value *xyz* must be specified in default length units. The normal vector will always be directed in the positive direction along the selected axis.

```
cut [at] x1 y1 z1 x2 y2 z2 x3 y3 z3
```

x1 y1 z1 x2 y2 z2 x3 y3 z3 Specifies three points P1 $= (x1, y1, z1)$, P2 $= (x2, y2, z2)$, and P3 $= (x3, y3, z3)$ that define a plane. The coordinate locations must be specified in default length units. The normal vector will be directed along the vector defined by the cross product $(\mathrm{P2} - \mathrm{P1}) \times (\mathrm{P3} - \mathrm{P1})$.

```
cut [at] point px py pz vector vx vy vz
```

`point` *px py pz* `vector` *vx vy vz* Point and normal specification of a plane. *px*, *py*, and *pz* define a point on the plane and the base of a vector normal to the plane. *vx*, *vy*, and *vz* complete the definition of the normal vector $\mathrm{N} = (vx - px, vy - py, vz - pz)$.

Note: When cutting planes are being used with 2D data (KDIM = 1), all subsets must specify '`k` *all*' rather than '`k` *1*'.

Examples

Examples of the use of the `cut` command can be found in the descriptions of the `genplot`, `genplot surface`, `integrate flux` and `integrate area` commands.

See Also: The `subset` and `surface` commands for how to define subsets; the `units` command forhow to define the default length unit; the `genplot` and `integrate area` commands for examples of use.

`crinkle` — Specify a crinkle plane

```
crinkle [at] {x|y|z} xyz |
crinkle [at] x1 y1 z1 x2 y2 z2 x3 y3 z3 |
crinkle [at] point px py pz vector vx vy vz
```

The `crinkle` command defines a crinkle plane that will be passed through all of the active subsets. The options, etc., are the same as described previously for the `cut` command.

`rake` — Specify locations for interpolation

```
rake x xbeg[,xend,xinc] y ybeg[,yend,yinc] z zbeg[,zend,zinc] |
rake line [begin] x1 y1 z1 [end] x2 y2 z2 [number] numpts |
rake plane [llhc] x1 y1 z1 [ulhc] x2 y2 z2 [lrhc] x3 y3 z3 [ndir] n [mdir] m |
rake polar [center] xc yc zc [begin] x1 y1 z1 [end] x2 y2 z2 -
   [rings [{equal|centroid} area]] numring [rakes] numrake -
   [hub [radius] r0] [angle {full|half|angle}] |
rake file filename [1D|2D|3D] |
rake cgd filename.cgd
```

The `rake` command specifies the interpolation points for the `interpolate` command. There are five possible formats for the `rake` command, as shown below. The first form of the command is used to generate a 2-D or 3-D rectangular grid of points. The second and third forms are used to generate a line or plane of points. The fourth form is used to generate a polar grid of points, like an engine face rake. The fifth form of the command is the most general and allows the user to define an arbitrary structured mesh of points.

Up to 128 rake commands may be specified. The rake list can be cleared with the `clear` command and displayed with the `show` command. All coordinate data must be entered in default length units.

```
rake x xbeg[,xend,xinc] y ybeg[,yend,yinc] z zbeg[,zend,zinc]
```

xbeg`[`*,xend,xinc*`]`, *ybeg*`[`*,yend,yinc*`]`, *zbeg*`[`*,zend,zinc*`]`
Specifies the beginning and ending points and increment of the points to be generated in each of the three Cartesian coordinates.

```
rake line [begin] x1 y1 z1 [end] x2 y2 z2 [number] numpts
```

`[begin]` *x1 y1 z1*	Specifies the first endpoint of the line.
`[end]` *x2 y2 z2*	Specifies the second endpoint of the line.
`[number]` *numpts*	The number of grid points that are to be equally spaced along the line from (*x1*,*y1*,*z1*) to (*x2*,*y2*,*z2*). The number includes the end points.

```
rake plane [llhc] x1 y1 z1 [ulhc] x2 y2 z2 [lrhc] x3 y3 z3 [ndir] n [mdir] m
```

`[llhc]` *x1 y1 z1*	The coordinates of the lower left hand corner of the plane.
`[ulhc]` *x2 y2 z2*	The coordinates of the upper left hand corner of the plane.
`[lrhc]` *x3 y3 z3*	The coordinates of the lower right hand corner of the plane.
`[ndir]` *n*	The number of grid points to be equally spaced along the line from (*x1*,*y1*,*z1*) to (*x2*,*y2*,*z2*). The number includes the end points.
`[mdir]` *m*	The number of grid points to be equally spaced along the line from (*x1*,*y1*,*z1*) to (*x3*,*y3*,*z3*). The number includes the end points.

```
rake polar [center] xc yc zc [begin] x1 y1 z1 [end] x2 y2 z2 -
   [rings [{equal|centroid} area]] numring [rakes] numrake -
   [hub [radius] r0] [angle {full|half|angle}]
```

`[center]` *xc yc zc*	Specifies the center point of a polar rake.
`[begin]` *x1 y1 z1*	Specifies an outer point of the first rake leg. This is used with the center to compute the radius of the rake that is used in the centroid area calculation.
`[end]` *x2 y2 z2*	Defines another point that, when combined with the `center` and `begin` points, specifies the plane in which the rake lies. It also defines the ordering of the rakes, which proceed in the direction defined by the vector going from the `begin` point to the `end` point.
`[rings [{equal|centroid} area]]` *numring*	Specifies the number of rings (points) to be generated. If the `equal area` or `centroid area` parameter is not specified, *numring* points will be uniformly distributed in the radial direction starting at the hub and proceeding to the outer radius. If the `equal area` parameter is specified, *numring* points will be located at the edge of $numring - 1$ annuli whose areas are all identical. In both cases there will be a point at the outer radius and the hub radius. If `centroid area` is specified, the points will be at the centroid of *numring* annuli, each of which as the same area.
`[rakes]` *numrake*	Specifies the number of rake legs to be generated. The rake legs are distributed uniformly in the circumferential direction. Includes the first and last legs. For a 360-degree rake the first and last legs are the same; thus nine legs will be spaced 45 degrees apart.
`[hub [radius]` *r0*`]`	The radius of the hub.
`[angle {full|half|`*angle*`}]`	The subtended angle of the rake.

```
rake file filename [1D|2D|3D]
```

filename	The name of a file that contains the interpolation points. The file defines one or more computational lines, planes, or volumes, each of which will become a zone in the output file created by the `interpolate` command. Single points may also be defined by creating a curve with only a single point. The format of the file is defined in Appendix B.
`[1D|2D|3D]`	The format of the rake file depends on the dimensionality of the data. See Appendix B for details.

```
rake cgd filename.cgd
```

filename.cgd	The name of a common grid (*.cgd*) file to be used as the rake file interpolation points. If the rake cgd file is unstructured, then the interpolated solution will be unstructured too.

Examples

A detailed example of the use of the `rake` command may be found in the description of the `interpolate` command.

See Also: The `interpolate` command for how to retrieve function information at specified spatial locations; the `units` command for how to define the default length unit.

`bledge` — Create an interpolation file of the boundary layer edge

```
bledge [surface] [maxdist dist] [normal] output intfile [overwrite]
```

dist	The maximum distance allowed for the edge of the boundary layer. Any points further from the surface will have their `iblank` value set to zero so they will not be displayed or used.
`normal`	The `normal` option causes two planes normal to the wall to be output. This is useful for programs that need to integrate normal to the wall.
intfile	The name of a file that the boundary layer interpolation points are written to. The file is ready to be used by the `interpolate` command using the `read` keyword.
`overwrite`	The `overwrite` option causes the output file to be overwritten if it exists. The default is to terminate if the output file exists.

The `bledge` command calculates the boundary layer edge, based on the method specified in the `boundary layer` command, in the current subset normal direction, and creates an interpolation surface of the boundary layer edge. The actual surface and the boundary layer surface are stored an interpolation file which may be used by the `interpolate` command to create a common file grid and flow file (*.cgf*).

Example

A detailed example of the `bledge` command may be found under the `interpolate` command.

See Also: The `interpolate` command for how to retrieve function information at specified spatial locations.

`blgrid` — Create a normal grid interpolation file

```
blgrid output intfile [npoints n] [distance d] [tandist first_spacing last_spacing]
```

intfile	The name of a file that the generated normal grid points are written to. The file is a 3D eagle file which can be used by the `interpolate` command.
`npoints` *n*	Causes *n* points to be placed on the lines generated normal to the surface. The default is 21.
`distance` *d*	The distance, in current length units, to grow normal to the surface. The default is 1.
`tandist` *first_spacing last_spacing*	A hyperbolic tangent distribution is placed on the lines normal to the surface, instead of an equal arc distribution, with the specified end spacing. If the spacing is zero, then spacing for that endpoint will float.

The `blgrid` command generates a volume grid from surface subsets by growing normal to the surface be the specified distance. This command is used to generate grids to calculate boundary layer properties such as displacement thickness. This is important when the original grid file is not very orthogonal to the surface since large errors in the boundary layer properties can occur in this situation. The volume grid is stored in a 3D eagle file which may be used by the `interpolate` command to create a common file grid and flow file (*.cgf*).

Example

```
grid fbdy.cgd
solution fbdy.cfl
units in
zone 1
surf j 1
blgrid output fbdy.egl npoints 11 distance 1.0 tandist 0.01 0.0
clear all
units mks                  ! Note that the units in the eagle file generated by
                           ! blgrid are mks
rake file fbdy.egl 3D
zone 1
interpolate cgf fbdy.cgf gridunits in     ! Make the grid units inches
                                          ! in the .cgf file
clear all
grid fbdy.cgf
solution fbdy.cgf
units in
zone 1
surf k 1                   ! All surfaces are k 1 in the new grid
var delta*                 ! Use default edge search criteria
list output delta.lis
```

See Also: The `interpolate` command for how to retrieve function information at specified spatial locations; the `rake` command for specifying rakes.

5 Variable Selection and Unit Control Commands

`variable` — Select variables to be processed

```
variable name [unit-override] [; name [unit-override] ...]
```

name	The name of a variable to be selected. The specified name must be present in the input files or must be one that can be derived from other information in the files. See the list below for the variables that can be derived.
unit-override	Specifies one or more unit-specifiers that may be specified on the unit command, a normalization quantity or a user specified scale and offset of the form: `[origin` *origin*`] [scale` *scale*`]`

The `variable` command is used to define the variables that are to be processed and optionally in what units of measure the data is to be presented. The unit-override specifications only affect the variable being selected; they do not alter the default units of measure.

`variable` commands accumulate, that is, each `variable` command adds to the list of variables that will be made available for subsequent processing. The list is passed on in the order that it was created. Up to 128 variables may be selected at any one time. The list of selected variables and their units of measure can be displayed with the `show` command and cleared with the `clear` command.

The following list represents variables that, if requested and not present in the input files, will be derived from other information in the files if sufficient information is available. Real gas effects are taken into account if the solution file has the necessary information.

Note that some variables require the specification of a surface, or normal direction into the flow field, and others require knowledge of "up" and "down" axes orientation. Variables denoted with (s) following their description indicate that the surface and normal direction into the flow field must be specified properly using the `subset` or `surface` command. Variables denoted with (o) are affected by the axes orientation set by the `orientation` command.

Table 1: Variables Available

name	**Description**
`x`	x coordinate
`y`	y coordinate
`z`	z coordinate
`dx`	Δx, measured from first point in subset
`dy`	Δy, measured from first point in subset
`dz`	Δz, measured from first point in subset
`ds`	Arc length, measured from first point in subset
`x/c`	Normalized x (only valid for GENPLOT files)
`y/b`	Normalized y (only valid for GENPLOT files)
`rho`	Density

Continued on next page

Table 1: **Variables Available** (*Continued*)

name	**Description**
`rho0`	Stagnation density
`p`	Pressure
`p0`	Stagnation pressure
`Cp`	Pressure coefficient
`Cpt`	Total pressure coefficient
`deltap`	Delta pressure
`pp`	Pitot pressure
`q`	Dynamic pressure
`T`	Temperature
`T0`	Stagnation temperature
`M`	Mach number
`Ma`	Equivalent isentropic Mach number
`Mt`	Turbulent Mach number
`a`	Speed of sound
`u`	x velocity
`v`	y velocity
`w`	z velocity
`V`	Velocity magnitude
`Vxy`	uv crossflow velocity magnitude
`Vyz`	vw crossflow velocity magnitude
`Vxz`	uw crossflow velocity magnitude
`e0`	Absolute stagnation energy per unit volume
`ek`	Kinetic energy per unit volume
`ei`	Sensible internal energy per unit volume
`h`	Sensible enthalpy
`h0`	Sensible stagnation enthalpy
`s`	Entropy
`S`	Strain rate magnitude
`Sxx, Sxy, Sxz`	Strain rate tensor components
`Syx, Syy, Syz`	Strain rate tensor components
`Szx, Szy, Szz`	Strain rate tensor components
`W`	Rotation rate magnitude or vorticity magnitude
`Wxx, Wxy, Wxz`	Rotation rate tensor components
`Wyx, Wyy, Wyz`	Rotation rate tensor components
`Wzx, Wzy, Wzz`	Rotation rate tensor components
`wx`	x component of vorticity
`wy`	y component of vorticity
`wz`	z component of vorticity
`swirl`	Swirl

Continued on next page

Table 1: Variables Available (*Continued*)

name	**Description**	
helicity	Helicity	
dila	Dilatation	
localpha	Local α (angle of attack)	*o*
locbeta	Local β (sideslip angle)	*o*
kappa	Thermal conductivity coefficient	
Q	Heat transfer rate	*s*
mu	Total viscosity coefficient	
mul	Laminar viscosity coefficient	
mut	Turbulent viscosity coefficient	
k	k from k-ϵ or SST turbulence model	
epsilon	ϵ from k-ϵ turbulence model	
omega	ω from SST turbulence model	
alphat	Turbulent thermal diffusivity	
var_ei, eps_ei	Variance and dissipation rate of internal energy	
var_h0, eps_h0	Variance and dissipation rate of total enthalpy	
var_h, eps_h	Variance and dissipation rate of static enthalpy	
delta	Boundary layer thickness	*s*
delta1 or delta*	Boundary layer displacement thickness	*s*
delta2 or THETA	Boundary layer momentum thickness	*s*
delta3	Boundary layer energy thickness	*s*
deltau	Boundary layer velocity thickness	*s*
THETAinc	Boundary layer incompressible momentum thickness	*s*
Cf	Skin friction coefficient	*s*
Cfx	x component of skin friction coefficient	*s*
Cfy	y component of skin friction coefficient	*s*
Cfz	z component of skin friction coefficient	*s*
tau	Total (laminar+turbulent) shear stress	*s*
taux	x component of shear total stress	*s*
tauy	y component of shear total stress	*s*
tauz	z component of shear total stress	*s*
uu, uv, uw	Turbulent stress components	*s*
vu, vv, vw	Turbulent stress components	*s*
wu, wv, ww	Turbulent stress components	*s*
uu+, uv+, uw+	Non-dimensional turbulent stress components	*s*
vu+, vv+, vw+	Non-dimensional turbulent stress components	*s*
wu+, wv+, ww+	Non-dimensional turbulent stress components	*s*
y+	Non-dimensional boundary layer wall coordinate	*s*
u+	Non-dimensional boundary layer velocity	*s*
k+	Non-dimensional turbulent kinetic energy	*s*

Continued on next page

Table 1: Variables Available (*Continued*)

name	**Description**	
`epsilon+`	Non-dimensional turbulent dissipation rate	*s*
`Rt`	Turbulent Reynolds number	
`Ry`	Turbulent Reynolds number based on y	*s*
`cmuRG`	C_μ, Rumsey-Gatski algebraic stress coefficient	
`fmuJL`	f_μ, Jones-Launder damping function	*s*
`fmuCH`	f_μ, Chien damping function	*s*
`fmuSP`	f_μ, Speziale damping function	*s*
`fmuLB`	f_μ, Lam-Bremhorst damping function	*s*
`C-`*species*	Mass fraction of *species*	
`X-`*species*	Mole fraction of *species*	
`Veff`	Effective collision frequency	
`MW`	Molecular weight	
`R`	Gas constant	
`cp`	Specific heat at constant pressure	
`cv`	Specific heat at constant volume	
`gamma`	Ratio of specific heats	
`hf`	Heat of formation	
`beta`	Effective gamma	
`Z`	Compressibility factor	

In addition to the above, for flows in a rotating reference frame (computed, for example, using Wind-US with the `ROTATE` keyword), the following variables may be requested for properties in the rotating reference frame: `p0r`, `T0r`, `Mr`, `ur`, `vr`, `wr`, `Vr`, and `e0r`. Note that the names are the same as the corresponding variable from the above list for the inertial (i.e., non-rotating) reference frame.

Also, due to the special significance of conservation variables, any *name* of the form *rho*xxxx* will be derived if sufficient information is present to derive *rho* and *xxxx*.

In addition to any of the *unit-specifiers* that may be specified on the `units` command, any one of the following normalization factors may be supplied:

Table 2: Variable Normalization Factors

unit-override	**Description**
`rhoinf`	Freestream density
`rho0inf`	Freestream stagnation density
`pinf`	Freestream pressure
`p0inf`	Freestream stagnation pressure
`ppinf`	Freestream pitot pressure
`qinf`	Freestream dynamic pressure

Continued on next page

Table 2: **Variable Normalization Factors** (*Continued*)

unit-override	**Description**
`Tinf`	Freestream temperature
`T0inf`	Freestream stagnation temperature
`hinf`	Freestream enthalpy
`h0inf`	Freestream stagnation enthalpy
`ainf`	Freestream speed of sound
`Minf`	Freestream Mach number
`uinf`	Freestream velocity
`muinf`	Freestream viscosity
`kinf`	Freestream k from the k-ϵ turbulence model
`einf`	Freestream ϵ from the k-ϵ turbulence model

The use of the `origin` and `scale` unit override allows the user to create their own units or nonrmalizations. The values specified for `origin` and `scale` are used as follows:

$$x' = (x - origin) \times scale$$

The user must be aware that if x is a dimensional quantity, *origin* and *scale* must take into account the fact that x is in SI units. A typical use would be the display of x normalized by the chord length.

Variable names may be preceeded by a minus sign to swap their sign. The minus sign is applied in SI units before any optional `origin`, `scale`, unit conversion, or normalization.

Example 1

```
! select pressure normalized by the freestream pressure
! also select Mach number
variable p pinf; M
! select u in feet/second (assume second is default time unit)
variable u feet
! select turbulent shear stress, swapping the sign
variable -rho*uv
! four variables have been selected!
```

Example 2

```
! select u velocity normalized by freestream velocity
variable u uinf
! enable English units
units english
! select v velocity in feet/second
variable v ft sec
! select w velocity in centimeters/second
variable w cm
```

Example 3

```
! select pressure in psi. Note that if one simply selected
! English units then the result would be in psf!
```

```
variable p lbf inches
! select heat transfer rate in BTU/(foot^2-hour)
variable Q feet hour BTU
```

Example 4

```
! Select Cp and x. x is normalized by the chord.
! The chord is 100 inches = 2.54 meters, therefore scale = 1/2.54 = .3937
! The leading edge was at 250 inches = 6.35 meters, therefore origin = 6.35
variable x origin 6.35 scale .3937
```

See Also: The `units` command for how to define units of measure; the `show` command for how to display the variable list; the `clear` command for how to clear the variable list; the `subset` and `surface` commands for how to specify the surface and normal direction into the flow field for variables marked with (s); the `orientation` command for how to specify the axes orientation for variables marked with (o); Appendix C for the formulas used for deriving variables.

`vector` — Define a vector variable

```
vector variable [unit-override]
```

variable	The name of a vector variable.
unit-override	A normalization quantity or one or more *unit-specifiers* that may be specified on the `units` command.

The `vector` command is used to specify the variable to be included in the plot file for generating a vector plot. The following vector variables are available:

Table 3: Vector Variables

variable	Description
`V`	Velocity
`Cf`	Skin friction
`W`	Vorticity

See Also: The `units` command for how to define units of measure; the `genplot surface` command for how to generate a file for vector plotting; the `plot vectors` command for how to perform vector plotting.

`units` — Specify default dimensional units of output data

```
units unit-specifier unit-specifier ...
```

unit-specifier The name of a unit or system of units.

The `units` command defines the default units of measure to be used for the subsequent presentation of data. CFPOST maintains units of measure for four basic units (mass, length, time and temperature), and two derived units (force and energy). All other units are derived from these basic units. For instance, units of pressure are derived from the force unit and the length unit. Similarly, velocity is derived from length and time units. *CFPOST uses SI units internally as its default system of units.* The default output system of units is FSS.

If a 'system of units' is specified as a *unit-specifier*, all six of the units of measure are set to the values defined for the selected 'system of units'. If a 'unit name' is specified as a unit specifier, only the corresponding unit of measure is modified.

Some commands produce geometry information with the geometry variables being explicitly requested (e.g.: `artis`, `plot3d`). On some other commands the user is providing locations, lengths, or areas (e.g.: `cut`, `integrate`, `rake`). In these situations, CFPOST needs to know the length unit in which the information is being given. The program does this by maintaining a *default length unit*. The default length unit is the length unit that is in effect just before the first variable is added to the variable list or, if no variables have been selected, at the time that a command that requires the default length unit is executed. Note that the first variable is either the first one added in an execution of CFPOST or the first one added after the variable list has been cleared.

Valid values for 'system of units' are:

Table 4: Systems of Units

unit-specifier	**Description**
`SI`	System International (meter, kilogram, second, degrees Kelvin, Newton, Joule)
`MKS`	A synonym for `SI`
`metric`	A synonym for `SI`
`CGS`	Centimeter, gram, second, degrees Kelvin, dyne and erg
`FSS`	Foot, slug, second, degrees Rankine, pound force, foot-pound force
`british`	A synonym for `FSS`
`english`	A synonym for `FSS`
`FPS`	Foot, pound mass, second, degrees Rankine, pound force, foot-pound force

Valid values for mass units are:

```
kg, kilogram
gram
slug
lb, lbm, pound
```

Valid values for length units are:

```
m, meter
cm, centimeter
mm, millimeter
ft, feet, foot
in, inch
```

Valid values for temperature units are:

```
K, Kelvin
C, Centigrade, Celsius
R, Rankine
F, Fahrenheit
```

Valid values for force units are:

```
N, Newtons
dynes
lbf (pounds force)
```

Valid values for energy units are:

```
Joule
erg
lbf-ft, ft-lbf
BTU
```

Example

```
! English system, but override energy unit
units english BTU
```

See Also: The `variable` command for how to override the current units of measure.

6 File and Report Generation Commands

`analyze` — Analyze/synthesize engine face data

```
analyze {engine|synthesis} output file [type T45|PW|GE]
```

`output` *file*	Specifies the name of the file that will contain the output from the analysis/synthesis.
`type T45|PW|GE`	Specifies the type of synthesis or the total pressure data to use for engine face calculations.

The `analyze synthesis` command generates a peak dynamic distortion pattern using the method defined by the `type` option. The data generated is stored in the *.cgf* file in a variable called `p0_`*type*, where *type* is `T45`, `PW`, or `GE`. The output file contains data similar to the `analyze engine` output, but tailored to the T45. Currently only the T45 methodology is supported and the peak distortion pattern is created using correlations established from the 26% T45 inlet database. The accuracy of the peak dynamic patterns is a very strong function of the accuracy of the steady-state or time-averaged total pressure pattern.

The `analyze engine` command creates a report of conditions and performance parameters for the specified rake using the total pressure data specified with the `type` option. If no `type` is specified then `p0` is used. The input for this command must have been created by a previous `interpolate` command with rakes specified by the `rake polar` command, or using the *cfcnvt* option "Convert ASCII rake to Common File rake CGF" (available in *cfcnvt* 1.40 and later) followed by an `analyze synthesis` command. If more that one rake was specified to `interpolate`, each rake will be analyzed and reported individually. If `rake file` was used to generate the *.cgf* file, then you must store the rake code 1.0 in zonal `fpar(40)` and the rake angle in zonal `fpar(42)` for this command to operate. The utility *fpro* has the capability to store these numbers.

Note: This command is not available for unstructured grids.

Example

```
grid cfd10.cgf
solution cfd10.cgf
unit in
zone 1
analyze synthesis type T45 output cfd10.lis
clear all
grid cfd10.cgf
solution cfd10.cgf
zone 1
analyze engine type T45 output cfd10dsyn.lis
```

See Also: The `rake polar` and `interpolate` commands for how to generate the input file to this command.

`artis` — Generate QWIKPLOT files for ARTIS

```
artis output file {formatted|unformatted|iris}
```

`output` *file*	Specifies the name of the QUIKPLOT file. If a file extension is not specified then "*.qpf*" will be assumed if `formatted` is specified, otherwise "*.qpd*" will be assumed.
`formatted`	Create the output file using FORTRAN formatted `WRITE` statements. The file must be translated into an unformatted file using the program *qputil* before it can be read by `artis`.
`unformatted`	Create the output file using FORTRAN unformatted `WRITE` statements. The file can be read only on the same type of system on which it was created.
`iris`	Create the output file so that it can be read directly by `artis` on a Silicon Graphics workstation. This option is *not* available on Cray computers.

The `artis` command generates files for use by the ARTIS (Aerodynamic Real-Time Imaging System) that runs on Silicon Graphics workstations.

The ARTIS program requires surfaces as input. `surface` commands may be used to define the surfaces or `subset` commands may be used to specify surfaces or groups of surfaces. Each surface will become a "section" in the output file. If the `subset` command is used to define the surfaces, the surface should be defined so that the surface normal vector would point towards the viewer if viewed from the "front" or "outside" of the surface.

Note: This command is not available for unstructured grids.

Example

```
units inches                 ! Specify default length unit for geometry
zone 1
   surface j 1 i 9,last      ! Omit the singular axis
zone 2
   surface j 1
   surface i last
variable Cp; M; T Tinf
artis output tma2 iris4
```

See Also: The `surface` command for how to define surfaces; the `subset` command on how to define surfaces and normal vectors; the `units` command on how the define the default length unit; the `variable` command on how to select variables and their units.

`calculate` — Calculate a new or predefined special function

```
calculate {reynolds [stress] bij value | prms | cpdot | -
   roughness height h | radeq emissivity e [tolerence tol] | -
   rel_humidity [freestream RHinf] | function function in postfix (HP Polish) form}
```

`reynolds [stress] b`*ij value*
: Calculates the Reynolds shear stress based on the input stress tensor b_{ij}. This tensor is symmetric so only the diagonal and lower diagonal elements are input and calculated. The equation solved is

$$R_{ij} = \rho k \left(\frac{2}{3} S_{ij} + 2B_{ij} \right)$$

where $-1/3 \le B_{ii} \le 2/3$, and $-1/2 \le B_{ij} \le 1/2$. The data is saved in variables named `R11`, `R22`, `R33`, `R12`, `R13`, and `R23`, and have units of N/m^2. Since these variables have known units all unit conversions are supported.

`prms`
: Calculates pressure fluctuations on a surface created with the `bledge` command. The pressure fluctuations on the surface are based on wall and boundary layer edge properties as follows:

$$0.006 \frac{\frac{1}{2}\rho_e U_e^2}{\frac{1}{2}\left(1 + \frac{T_w}{T_e}\right) + \frac{1}{10}(\gamma - 1)M_e^2}$$

where the subscripts e and w denote edge and wall properties, respectively. The variable stored is called `prms`.

`cpdot`
: Calculates C_p dotted with the surface normal vector as well as lift, drag, and side force. This function only works on surface grids, thus you must use the `copy` command to copy out your walls. The variables stored in the file have the following names: `Cp*Nx`, `Cp*Ny`, `Cp*Nz`, `Cp*Ndrag`, `Cp*Nlift`, and `Cp*Nside`.

`roughness height` *h*
: Calculates, for each wall, variables that estimate drag changes due to protruding objects of height h. The variables are dynamic pressure and Reynolds number based on height h respectively, integrated normal to the walls to the height h specified. The walls are determined from the actual boundary conditions in the grid file. The variables stored in the file have the following names: `qh` and `rh`.

`radeq emissivity` *e* `[tolerence` *tol*`]`
: Calculates the radiation equilibrium temperature based on the wall and boundary layer edge properties. The emissivity of the gas is supplied along with an optional tolerance to converge the iterative proceedure (default 0.001). The input file must be an interpolation *.cgf* file generated using the `bledge surface normal` and `interpolate` commands.

`rel_humidity [freestream` *RHinf*`]`
: Calculates the local relative humidity in the air based on the

freestream relative humidity and local pressure and temperature. The variable stored in the file is called `RHl` and is calculated as follows:

$$\texttt{RHl} = RHl\frac{p}{p_\infty} \times 10^{\frac{\frac{-2263}{T_\infty}\left(\frac{T}{T_\infty}-1\right)}{\frac{T}{T_\infty}}}$$

`function` *function in postfix (HP Polish) form*
Calculates a user defined function. The user defined functions have the form

`funnam =` *var1 var2 opr1 var3 opr2* `...`

where the *var*s denote variables and the *opr*s denote operators. The function is calculated as

`funnam = ( ... (`*var1 opr1 var2*`)` *opr2 var3*`) ... )`

The `calculate` command gives the user the ability to use one of the built in functions or to define their own user function. Predefined functions in general are complex functions which are not just simple combinations of other variables. All functions are calculated and stored as a predefined or user defined variable in the *.cfl* file. Once the variable is saved in the file it may be operated on like any other variable except that automatic unit conversions are not supported for user defined functions. The function is calculated in all zones specified with the `zone` command on the entire zone, regardless of any subset specified on that zone.

Note that since the calculated function is written into the *.cfl* file, for CFPOST versions 4.1 and later the `solution` command must use the `mode` option to open the file in read-write mode.

For user defined functions all data is calculated and stored in the metric system. Since the variable units are not known standard unit conversions are not supported, but may be performed using the `scale` option in the `variable` command. Thus any constants which are input as part of the function must be in metric units. All CFPOST supported infinity conditions (`pinf`, `Tinf`, etc.; see Table 2 starting on p. 34) may be used in the equation as well as the constants `gamma`, `R`, `Pr`, and `Prt`. The values stored in the *.cfl* file for these constants will be used in the calculation. Remember that the variable names and predefined constant names are case sensitive. The operations available are `+`, `-`, `*`, `/`, `**`, `sinh`, `cosh`, `tanh`, `sin`, `cos`, `tan`, `atan`, `atan2`, `log`, `ln`, `exp`, `erf`, `erfi`, `min`, `max`, and `abs`. All variables and operators in the equation must be separated by white space. For single operator functions like `sin`, a dummy argument must be supplied (see following examples).

Warning: If you are writing over an existing variable which has reference and scaling data, you must renormalize the data in the function definition since the function is overwritten into the file as specified and reference and scaling data is not modified.

The following examples assume that the CFPOST version being used is 4.1 or later, and thus the *.cfl* file must be opened in read-write mode.

Example 1

Calculate the Reynolds shear stress tensor using the stress tensor shown below, and list it to a file.

$$B = \begin{vmatrix} 0.1 & -0.5 & 0.0 \\ -0.5 & -0.06 & 0.01 \\ 0.0 & 0.01 & -0.1 \end{vmatrix}$$

```
solution testke.cfl mode read_write
zone 1
calculate reynolds b11 +0.1 b22 -0.06 b33 -0.1 b12 -0.5 b13 0.0 b23 0.01
clear variable
units english
variable R11;R22;R33;R12;R13;R23
list output test.lst
```

Note that the variable list is cleared after the `calculate` command since it sets the variable list to the variables required to calculate the function.

Example 2

Calculate the normalized turbulent kinetic viscosity (`nut`) and plot the variable on a cut at $z = 0.0$. The functions are defined using standard CFPOST variables as `nut = mut/rho` and `nutinf = muinf/rhoinf`.

```
solution test.cfl mode read_write
zone 1 to last
calculate function nut = mut rho / muinf / rhoinf *
clear variable
variable nut
unit inches
cut at z 0.0
plot color contours
```

Note that:

```
nut/nutinf = (mut/rho)/(muinf/rhoinf)
           = (mut/rho)*(rhoinf/muinf)
           = (mut/rho/muinf)*(rhoinf)
           = mut/rho/muinf*rhoinf
```

Thus the postfix form of the equation is one which can be written without parentheses and is performed in left to right order.

Example 3

Calculate the corrected airflow at each point weighted by the total area, average the results in units of lb_m/s and write to a file. The function is defined using standard CFPOST variables as follows:

```
CMF = rho*A*V*(T0/Tref)**0.5/(p0/pref)
```

where `A` is the total area, `Tref` = 518.7 °R, and `pref` = 14.7 psi are fixed constants. Since all constants must be input to the function in metric units these constants become `Tref` = 270.39 K and `pref` = 101353.0 P. We assume the area is a known `A` = 1.5 m^2.

```
solution test.cfl mode read_write
zone 5
   surface I l
calculate function CMF = t0 270.39 / 0.5 ** rho * V * 1.5 * 101353.0 * p0 /
clear variable
variable CMF scale 2.204
list average output cmf.lis
```

Note that the `scale` in the `variable` command above converts the variable `CMF` from the metric system (kg/s) to the english system (lb_m/s).

Example 4 — Using the sine function

```
calculate function fsine = a 1.0 sin     ! gives sin(a), the 1.0 is a dummy
```

Example 5 — Using the tangent function

`calculate function` *ftan = a b tan2*   `!` *gives tan2(a,b)*

See Also: The `subset` command on how to define surfaces and normal vectors; the `units` command on how to define the default length unit; the `variable` command on how to select variables and their units; the `clear` command for clearing subsets and variables; the `list` command for listing variables; the `plot` command for plotting data.

`cfdfem` — Generate a CFDFEM file for CGSA

```
cfdfem output file
```

`output` *file*	Specifies the name of the file to receive the data. A file extension of "*.fem*" will be provided if none was supplied.

The `cfdfem` command produces a file for use in Boeing structural analysis.

The subsets must specify surfaces. `surface` commands may be used to define the surfaces or `subset` commands may be used to specify surfaces or groups of surfaces. If the `subset` command is used to define the surfaces, the surface should be defined so that the surface normal vector would point towards the viewer if viewed from the "front" or "outside" of the surface.

If a `variable` command is not specified, a standard CFDFEM file will be produced which will contain the geometry and pressure coefficient. The units of the geometry information will be the default length unit. If a `variable` command is specified then the geometry and the selected variables will be written to the file. Such a file will probably be usable only by a custom program.

The orientation of the axes as determined by the `orientation` command determine the order of appearance of the x, y, and z data in the output file. The first coordinate is the "side" or buttline coordinate, the second is the downstream or fuselage station coordinate and the last is the up or waterline coordinate.

Note: This command is not available for unstructured grids.

Example

```
units inches                 ! Specify the default length unit for the geometry
zone 1
   surface j 1 i 9,last      ! Omit the singular axis
zone 2
   surface j 1
cfdfem output tma2
```

See Also: The `subset` command on how to define surfaces and normal vectors; the `units` command on how to define the default length unit; the `variable` command on how to select variables and their units.

`copy` — Copy or append to common files

```
copy [grid [to] cgdfile] [solution [to] cflfile] [both [to] cgffile] [append]
```

`grid [to]` *cgdfile*	The name of the common file to receive the variables from the current grid file. This parameter must not be specified if `both` is specified.
`solution [to]` *cflfile*	The name of the common flow file to receive the variables from the current solution file. This parameter must not be specified if `both` is specified.
`both [to]` *cgffile*	The name of the common grid and flow file to receive the variables from both the current grid and solution files. This parameter must not be specified if either `grid` or `solution` is specified.
`append`	Indicates the created zones are to be appended to the existing output files. If this qualifier is not specified then the output files must not exist prior to the execution of this command.

The `copy` command is used to create new common files from data extracted from the input files. This command is useful for extracting pertinent data from large CFD solutions for transmittal to other locations for other post-processing or for re-partitioning an existing solution due to memory constraints.

Each subset becomes a zone in the output file(s). All variables from the input grid and solution files are copied to their respective output files (i.e., any `variable` commands are ignored). Global information from the input file (flow conditions, reference and scaling data, etc.) is copied to directly to the output file. A zone in the output file inherits all zone information from the source zone in the input file. Boundary condition and zone coupling information is *not* transferred to the output file, so the boundary conditions must be reset if the output file is going to be used by a flow solver. For unstructured grids only unstructured surfaces can be copied, the interior cannot be.

Example

A two-zone solution was started and it was determined that first zone was too dense in the k direction, and the second needed to be split into two zones so it would require less memory.

```
grid test2z.cgd
solution test2z.cfl
zone 1
   subset i all j all k all,2    ! Every other point in K
zone 2
   subset i 1,51 j a k a         ! This will be the new zone 2
   subset i 52,last j a k a      ! This will be the new zone 3
copy grid to test3z.cgd solution to test3z.cfl
! Be sure to reset the boundary conditions and recouple after performing this operation!!!
```

`delta` — Create a delta *.cfl* file

```
delta [difference] cfl cflfile output file [append]
```

`cfl` *cflfile*	Specifies the name of the *.cfl* file to be subtracted from the current solution file.
`output` *file*	Specifies the name of the file to contain the delta *.cfl* data.
`append`	Indicates the created zones are to be appended to the existing output files. If this qualifier is not specified then the output files must not exist prior to the execution of this command.

The `delta` command is used to create a *.cfl* file containing the difference between two solutions. The difference is just point to point, there are no projections performed, thus only the zone dimensions must match between the two solutions. Deltas will be computed for all variables specified with the `variable` command or the default of all variables in the *.cfl* file will be delta'ed. The reference conditions in the output *.cfl* file will be the same as the *.cfl* file specified with the `solution` command. All output variables will be written in the metric system of units.

Warning: You cannot derive new variables from delta variables, you must specify all desired variables in the `variable` command before performing the `delta` command.

Example

```
solution base.cfl
zone 1
   surface j 1
zone 2
   surface u 3
copy grid to delta.cgd
variable Cp;p;M
delta difference cfl new.cfl output delta.cfl
```

See Also: The `subset` command on how to define surfaces and normal vectors; the `variable` command on how to select variables and their units; the `copy` command for copying subsets to another grid or flow file.

`genplot` — Create *x*-*y* plot files

```
genplot output file [multiple plots|multiple variables] -
   [multiple segments|merge_segments] [blanking|noblanking] [overwrite]
```

`output` *file*	Specifies the name of the file to contain the plot data. A "*.gen*" file extension will be provided if none was supplied. This is a text file and it may be freely moved between different computer systems.
`multiple plots`	Indicates that each dependent variable will be plotted individually. This is the default value.
`multiple variables`	Indicates that all dependent variables will be plotted on the same plot.
`multiple segments`	Forces all segments to be written separately.
`merge_segments`	Merges segments into one segment. Used to merge segments generated by a cutting plane across multiple zones.
`blanking`	Indicates that grid blanking data (if it exists) should be used to skip hole points.
`noblanking`	Indicates that grid blanking data should be ignored.
`overwrite`	Causes the output file to be overwritten with the new data. The default is not to overwrite.

The `genplot` command generates *x*-*y* plot files for use by the `plot` command. The `variable` command defines the variables that are to be included in the output file. The first variable will be the independent variable for all plots and will appear on the *x*-axis. The second and subsequent variables will be the dependent variables and will always occur on the *y*-axis. There must always be at least two variables active at the time of the `genplot` command.

The data to be output will be curves. One way to explicitly define a curve is with the `subset` command. The second method is to define a cutting plane with the `cut` command that will intersect a surface defined with the `subset` command.

For unstructured grids, a cutting plane on a surface is the only way to create a GENPLOT file.

Example 1

Generate a plot file for C_p versus x.

```
zone 1
   subset i all j 1 k 15
variable x inches; Cp
genplot output pltfile
```

Example 2

Just like the previous example except use a cutting plane to generate the curve rather than plotting along a grid line on the surface.

```
zone 1
   surface j 1    ! Define the surface to be cut
units inches      ! Specify default length unit
cut at z 150.0    ! Constant z cutting plane
```

```
    variable x inches; Cp
    genplot output pltfile
```

Example 3

An unstructured example:

```
    zone 1
       surface u 7     ! Define the surface to be cut
    units inches       ! Specify default length unit
    cut at y 20.0      ! Constant y cutting plane
    variable x inches; M
    genplot output pltfile
```

See Also: The `subset` and `surface` commands on how to define curves and surfaces; the `cut` command on how to define cutting planes; the `variable` command on how to select variables and their units; the `plot` command on how to plot the data.

`genplot surface` — Create surface files for contour plotting

```
genplot surface output file [overwrite]
```

`output` *file* Specifies the name of the file that will contain the plot data. A file extension of "*.gpc*" will be provided if none was supplied. This file contains binary information and may be interchanged only between compatible computers.

`overwrite` Causes the output file to be overwritten with the new data. The default is not to overwrite.

The `genplot surface` command generates an unstructured grid file for use by the `plot contours` command or any other program that can process an unstructured grid. The file contains geometry information and the variables. Variables may be specified by the `variable` command or defaulted to all variables in the *.cfl* file. The units of the geometry and variables will be in the metric system.

A surface is defined as the intersection of a plane defined with the `cut` command and one or more volumes defined with `subset` commands. Each surface will be written into an unstructured zone as a unique unstructured surface for each zone intersected. Each surface will be individually displayed when processed by the `plot contours` command.

Note: When using `cut` with a 2-D solution (KDIM = 1), always specify "`k` *all*" rather than "`k` *1*" in all `subset` commands.

Example 1

Generate a plot file of Mach number and normalized static pressure for several fuselage station cuts.

```
subset i all j all k all    ! Specify a default subset
zone 1
zone 2
units inches                ! Specify default length unit
cut at x 50.0               ! Constant FS cutting planes
cut at x 75.0
cut at x 100.0
variable M; p pinf
genplot surface output pltfile overwrite
```

Example 2

An unstructured example.

```
subset u all      ! Specify a default subset
zone 1
zone 2
units inches      ! Specify default length unit
cut at x 10.0     ! Constant FS cutting planes
variable M; Cp
genplot surface output pltfile
```

See Also: The `subset` command on how to define volumes; the `cut` command on how to define a cutting planes; the `variable` command on how to select variables and their units; the `plot contours` command on how to plot the data.

`integrate` — Perform area-weighted, mass-weighted, or mass-flux-weighted integration

```
integrate [centroid] {area|mass|massflux} output listfile [axisymmetric] -
   [blanking|noblanking] [plot [x|y|z] plotfile]
```

`centroid`	Indicates that the plot file (if requested) will also contain a plot of the computational plane area and an approximation of the centroid normal area.
`area`	Indicates that area-averaged integration is to be done.
`mass`	Indicates that mass-averaged integration is to be done.
`massflux`	Indicates that mass-flux-averaged integration is to be done.
listfile	The name of the file to contain the printed results of the integration. A file extension of "*.lis*" will be provided if none was specified.
`axisymmetric`	Indicates that the files represent a 2D axisymmetric solution. The solution is assumed to be fully symmetric (360 degrees) about the x axis. Results must be divided appropriately if not fully symmetric.
`blanking`	Indicates that hole points (if defined in the grid file) will be omitted from the integration.
`noblanking`	Indicates that blanking data will not be interrogated for the presence of grid holes. All selected points will be included in the integration.
`plot [x|y|z]` *plotfile*	Indicates that a plot file of the area average values is to be produced. The area averaged value on a surface will appear on the y axis and the average value of the selected coordinate (`x`, `y`, or `z`) will appear on the x axis. This option is only useful if the subsets select a group of planes, or you have a number of parallel cutting planes.

Each of the currently selected variables is integrated over each of the surfaces or cutting planes in the current subsets. The output file contains a listing of the result of the integration on each of the surfaces. The output file includes the area of the surface and the area weighted average, mass or mass flux weighted average, average, standard deviation, minimum value and maximum value for each of the specified variables. If a plot file is requested then a curve will be generated where each point represents either the area or mass or mass flux averaged value (depending on which is selected) of a variable on a surface.

For a surface defined with a `surface` or `subset` command, the perimeter of the area to be integrated is defined by the grid lines at the extremes of the subset. The interior of the region is viewed as a collection of independent polygons bound by grid lines. A surface defined with a cutting plane is also an independent collection of polygons, created by intersecting the plane with the grid lines in the subsets. In either case, each polygon is processed independently and added to the total. The value of a variable in a cell is considered to be the average of the values at the verticies.

Note: When using this command with a 2-D solution (KDIM = 1), always specify "`k` *all*" rather than "`k` *1*" in all `subset` commands.

Note: For unstructured grids the surface(s) to be integrated must be defined with the `surface` command.

Note: CFPOST integrates on the cell vertex, whereas Wind-US integrates on the cell center. These methods are equivalent when integrating the entire surface, but small differences will occur when integrating on a subset of a zone surface.

Example 1

Determine the area averaged total pressure recovery and Mach number at the engine face (assume `i` *last* is the engine face):

```
zone 4
   subset j all k all i last
variable p0 p0inf; M
integrate area output areaint
```

Example 2

Maybe the engine face was not at a constant *i*-plane! Use the `cut` command to specify the engine face plane.

```
zone 4
   subset j all k all i all     ! Select a bigger subset
units inches                    ! Set default length unit
cut at x 150.0                  ! Engine face location in inches
variable p0 p0inf; M
integrate area output areaint
```

Example 3

Generate a plot file of total pressure recovery, normalized pressure and Mach number versus the *x* coordinate in a duct. Note that all *i*-planes are selected. The plot file *intplot* would be plotted with the `plot` command.

```
zone 4
   subset j all k all i all
variable p0 p0inf; p pinf; M
integrate area output intlist plot x intplot
```

Unstructured Example 1

```
zone 4
   subset u 3
variable p0 p0inf; M
integrate area output areaint
```

Unstructured Example 2

```
zone 4
   subset u 200 370       ! Select a bigger subset
units inches              ! Set default length unit
cut at x 150.0            ! Engine face location in inches
variable p0 p0inf; M
integrate area output areaint
```

See Also: The `subset` and `cut` commands for how to define surfaces; the `variable` command for how to select variables and their units; the `plot` command for how to plot data; Appendix C for equations.

`integrate flux` — Integrate fluxes through a surface

```
integrate flux output listfile [axisymmetric] [iviscous] -
   [blanking|noblanking] [ptbypt] [consaverage] [plot [x|y|z] plotfile]
```

listfile	The name of the file to contain the printed results of the integration. A file extension of "*.lis*" will be provided if none was specified.
`axisymmetric`	Indicates that the files represent a 2D axisymmetric solution. The solution is assumed to be fully symmetric (360 degrees) about the x axis. Results must be divided appropriately if not fully symmetric.
`iviscous`	Indicates only those cells where the velocity at all vertices of the cell is non-zero are included in the integration. This is useful where part of the surface may be a wall.
`blanking`	Indicates that hole points (if defined in the grid file) will be omitted from the integration.
`noblanking`	Indicates that blanking data will not be interrogated for the presence of grid holes. All selected points will be included in the integration.
`ptbypt`	For each cell or face on the surface, include a point-by-point list of the coordinates of the cell or face center, its area, and the local mass-flux values.
`consaverage`	Use conservation averaging when computing integrated fluxes
`plot [x|y|z]` *plotfile*	Indicates that a plot file of mass, momentum, and energy flux is to be produced. The mass, momentum, and energy flux will appear on the y axis and the average value of the selected coordinate (`x`, `y`, or `z`) will appear on the x axis. This option is only useful if the subsets select a group of planes, or you have a number of parallel cutting planes.

The `integrate flux` command integrate the fluxes through surfaces. Mass flux, momentum flux, pressure flux and gross thrust are displayed for each surface. A total of all surfaces will be included at the end of the listing.

A surface may be defined directly with a `surface` command (for structured or unstructured grids), a `subset` command (structured grids only), or as the intersection of a cutting plane defined by the `cut` command. If a `subset` command is used to define a surface, it should be defined so that the surface normal vector would point into or out of the volume as desired. The same consideration should also be given when defining cutting planes.

For a surface defined with a `surface` or `subset` command, the perimeter of the area to be integrated is defined by the grid lines at the extremes of the subset. The interior of the region is viewed as a collection of independent polygons bound by grid lines. A surface defined with a cutting plane is also an independent collection of polygons, created by intersecting the plane with the grid lines in the subsets. In either case, each polygon is processed independently and added to the total. The value of a variable in a cell is considered to be the average of the values at the verticies.

Note: When using this command with a 2-D solution (KDIM = 1), always specify "`k` *all*" rather than "`k` *1*" in all `subset` commands.

Example

Generate a plot file of mass flow, momentum, and energy versus the x coordinate in a duct. Note that all i-planes are selected. The plot file *intplot* would be plotted with the `plot` command.

```
zone 4
   subset j all k all i all
integrate flux output intlist plot x intplot
```

See Also: The `surface` command for how to define surfaces; the `subset` command for how to define surfaces and normal vectors; the `units` command for how to define units.

`integrate force` — Integrate forces on a surface

```
integrate force output listfile [axisymmetric] [iviscous] [noviscous] -
   [first order] [ipinf] [reference area area] -
   [reference moment xm ym zm] [reference length length] -
   [reference b span] [reference cbar mean-aerodynamic-chord] -
   [blanking|noblanking]
```

listfile	The name of the file to contain the printed results of the integration. A file extension of "*.lis*" will be provided if none was specified.
`axisymmetric`	Indicates that the files represent a 2D axisymmetric solution. The solution is assumed to be fully symmetric (360 degrees) about the x axis. Results must be divided appropriately if not fully symmetric.
`iviscous`	Indicates that only those cells whose corners all specify zero velocity are to be included in the integration. This is useful for surfaces that are only partially walls.
`noviscous`	Do not compute and add the viscous terms to the force. Only the pressure forces are included.
`first order`	Indicates that normal derivatives are to be computed first order rather than the default second order.
`ipinf`	The pressure terms are to be integrated as $p\,dA$ instead of $(p - p_\infty)\,dA$.
`reference area` *area*	The reference area to be used for calculation of lift and drag coefficients.
`reference moment` *xm ym zm*	Specifies the x, y, and z locations about which moments will be calculated.
`reference length` *length*	The reference length to be used for normalizing moments. This should not be specified if `reference b` and `reference cbar` are specified.
`reference b` *span*	Specifies the normalizing component for the non-pitching axis moments. This should not be used if `reference length` is specified. For symmetric cases you must supply the semispan.
`reference cbar` *mean-aerodynamic-chord*	Specifies the normalizing component for the pitching axis moment. This should not be used if `reference length` is specified. The default is the reference length.
`blanking`	Indicates that hole points (if defined in the grid file) will be omitted from the integration.
`noblanking`	Indicates that blanking data will not be interrogated for the presence of grid holes. All selected points will be included in the integration.

The `integrate force` command integrates the pressure and viscous forces and heat transfer rates on a surface and produces a report of the the integrated values, moments and lift and drag coefficients. An individual report is provided for each surface as well as for the sum of the contributions for all surfaces. Only pressure terms are integrated for unstructured surfaces.

`surface` commands (for structured or unstructured grids) or `subset` commands (structured grids only) must be used to define the surfaces. If `subset` commands are used they should be defined so that the surface normal vector is directed into the volume.

The perimeter of the area to be integrated is defined by the grid lines at the extremes of the subset. The interior of the region is viewed as a collection of independent polygons bounded by grid lines. Each polygon is processed independently and added to the total. The value of a variable in a cell is considered to be the average of the values at the vertices. Forces are assumed to act at the centroid of the polygon.

Note: When using this command with a 2-D solution (KDIM = 1), always specify "`k` *all*" rather than "`k` *1*" in all `subset` commands unless `noviscous` has been specified and a `set zcoordinate` command has been issued.

Structured Example

```
zone 1
   surface j 1 i 9,last    ! Omit the singular axis
zone 2
   surface j 1
integrate force output intfrc iviscous -
      reference length 30.0 -
      reference area 100.0 -
      reference moment 15.0 0.0 1.5
```

Unstructured Example

```
zone 2
   surface u 5
zone 3
   surface u 3
integrate force output intfrc iviscous -
      reference length 70.0 -
      reference area 300.0 -
      reference moment 1.0 25.0 15.0
```

See Also: The `surface` command for how to define surfaces; the `subset` command for how to define surfaces and normal vectors; the `units` command for how to define units.

`integrate volume` — Integrate forces and fluxes on volume faces

```
integrate volume output listfile [detail] [axisymmetric] [iviscous] [noviscous] -
   [first order] [reference area area] -
   [reference moment xm ym zm] [reference length length] -
   [reference b span] [reference cbar mean-aerodynamic-chord] -
   [blanking|noblanking] [ptbypt] [consaverage]
```

All parameters are identical to the parameters for the `integrate flux` and/or `integrate force` commands, with the exception of the following:

`detail`	Indicates that a detailed report (of about 60 lines) is to be provided for each surface specified. If omitted, a one line summary for each surface will be provided.
`ipinf`	`ipinf` is automatically selected for `integrate volume`.

The `integrate volume` command produces a report on the degree of conservation of mass, momentum and energy within a control volume. The report gives a summary of the forces and fluxes on each face of the control volume as well as a summary of conservation resulting from adding the contributions from each face in the volume. Each volume will be reported independently.

This command calls the `integrate flux` and `integrate force` commands internally. Thus, all of the information supplied in the description of those commands is applicable to this command.

Note: This command is not implemented for unstructured grids.

Example

Get a summary for all zones in a three-zone solution.

```
grid test.cgd
solution test.cfl
! Define a default control volume
surface i 1
surface i last
surface j 1
surface j last
surface k 1
surface k last
zone 1
zone 2
zone 3
integrate volume iviscous output intvol
```

See Also: The `integrate flux` and `integrate force` commands; the `surface` and `subset` commands for how to define surfaces.

`interpolate` — Interpolate to rake point locations

```
interpolate [cgf cgffile] [write savefile] [read savefile] [tolerance tol] -
   [gridunits unit]
```

`cgf` *cgffile*	Specifies the name of a common file to receive the results of the interpolation request. This file may then be used by other CFPOST commands such as `list` and `plot` to display the results in the manner you desire.
`write` *savefile*	Specifies the name of an interpolation save file to be created. The interpolation data (zone, cell, location in cell) for the currently selected rakes and grid is written to this file for later use by the `read` *savefile* parameter.
`read` *savefile*	Specifies the name of an interpolation save file created by an earlier execution with the `write` *savefile* parameter. In this case, any currently selected rakes are ignored and the interpolation information is used from *savefile*. This greatly speeds up the interpolation process when interpolating at the same points in multiple solutions run on the same grid since the search for the cells containing the selected points is not required.
`tolerance` *tol*	The maximum allowable distance (in current units) from an interpolated point to the bounds of the subset. The default value is 0.001.
`gridunits` *unit*	Specifies the units to use when storing the grid in the *.cgf* file. Note that solution variables are still stored in the MKS system.

The `interpolate` command is used to retrieve information from the input files at specified Cartesian coordinates rather than computational (i, j, k) coordinates. The command would most often be used to extract data at experimental probe point locations. The `rake` command is used to specify the locations of the points for which information is desired. This command may also be used to interpolate onto a surface. If only surface subsets are specified, the program will perform a bilinear interpolation on the surface. This is very useful for extracting values at experimental tap locations or for interpolating a solution on one grid onto a different grid. For unstructured grids only volume interpolation is supported.

The output from this command is another common file that contains the specified x, y, and z locations and the interpolated values of the selected variables organized into a structured grid format. If no variables have been selected, all variables from the input files are copied to the output file. The information for each `rake` command is written into its own zone with the I dimension being the number of x points, the J dimension being the number of y points, and the K dimension being the number of z points. The x, y, and z coordinates are sorted into ascending order along each of the coordinate directions.

The variables are written to the file in metric units, and all reference conditions are copied. Note that if the original solution data was stored on cell centers, then the interpolation factors are calculated on cell centers.

Example 1

```
grid test.cgd
solution test1.cfl
units inches          ! set default length unit for rake input
! rake 1, dimension 101 x 20 x 1
rake x 120.2 130.3 0.1 y 50.5 60.5 0.5 z 82.0
```

```
! rake 2, dimension 101 x 40 x 1
rake x 120.2 130.3 0.1 y 50.5 60.5 0.5 z 82.0
! rake 3, dimension 201 x 80 x 1
rake x 120.2 130.3 0.1 y 50.5 60.5 0.5 z 82.0
! rake 4, dimension 13 x 1 x 1
rake line begin 1.0 6.2 8.3 end 1.5 6.5 9.2 num 11
! select variables to be interpolated
variable M;p;T;u;v;w
! interpolate variables and write a save file for a later
! interpolation on a different solution with the same grid
interpolate write test.int cgf test1.cgf
! now we want to list the results
clear all            ! reset CFPOST
grid test1.cgf       ! select new file as the grid file
solution test1.cgf   ! also select it as the solution file
! select the variables to list
variable x inches; y inches
variable M
variable locbeta     ! locbeta is derived for u and w
! select the subsets
subset i a j a k a ! create a default subset
zone 1               ! zone 1 contains data from rake 1
zone 2               ! zone 2 contains data from rake 2
zone 3               ! zone 3 contains data from rake 3
zone 4               ! zone 4 contains data from rake 4
list output test1    ! test1.lis contains the results
```

Example 2

```
! Use interpolation save file to get the data from the
! same points in a new solution
solution test2.cfl
variable M;p;T;u;v;w
! Use the interpolation save file created in example 1
interpolate read test.int cgf test2.cgf
```

Example 3

```
units inches
! Specify rake locations using a file
rake file taps.egl
zone 1
   surface j 1
zone 2
   surface k 1
! Since only surfaces are specified interpolation is bilinear on the surface
interpolate cgf taps.cgf tolerance .01
```

Example 4

This example uses the `bledge` command to generate an interpolation file, performs the interpolation, and then uses the `calculate` command to calculate pressure fluctuations on the boundary layer edge.

```
grid 3post.cgd
solution 3psau.cfl
units inches
zone 2
   surface j last
zone 4
   surface j last
zone 5
   surface j last
zone 7
   surface j last
zone 8
   surface j last
boundary layer is y+ of 300
bledge surface output bledge.int
clear all
interpolate read bledge.int cgf bledge.cgf
clear all
grid bledge.cgf
solution bledge.cgf
zone 1
   subset i a j 30,last k 1,2
zone 2
   subset i a j 30,last k 1,2
zone 3
   subset i a j a k 1,2
zone 4
   subset i a j a k 1,2
zone 5
   subset i a j a k 1,2
units metric
clear variable
calculate prms
clear variable
variable M;p pinf;T Tinf;q qinf;prms
plot3d q bledgeq.plt iris
```

See Also: The `bledge` command for how to specify the interpolation points; the `boundary layer` command for specifying how to find the boundary layer; the `subset` command for how to define the domain to be searched; the `variable` command for how to select variables and their units; the `clear` command for how to clear subsets and variables; the `calculate` command for calculating special functions; the `plot3d` command to create a plot3d file.

`list` — List data to screen or a file

```
list [output file] [limits] [average] [[inside|outside] range low [to] high] -
   [blanking|noblanking] [pause|nopause] [lines lines-per-page] [raw] -
   [[nowrap|wrap linelength]
```

`output` *file*	Specifies the name of a file to contain the resultant listing. If no file is specified the output will go to the terminal. A default file extension of "*.lis*" will be provided if a file name is supplied without an extension.
`limits`	Displays the minimum and maximum value and their locations for each variable. If more than one subset is specified, the values will be the minimum and maximum values of all the zones.
`average`	Displays the average value for each variable. If more than one subset is specified, the values will be the average values of all the zones.
`inside|outside`	List only values inside or outside the range specified by the `range` keyword. The default value is `inside`.
`range` *low* `[to]` *high*	Limits the display of variables to those locations where the value of the first variable in the variable list is in(out) of the range *low* and *high*. If `inside` is specified or implied, only locations where the value is between *low* and *high* inclusive will be displayed. If `outside` is specified, only locations where the value is less than *low* or greater than *high* will be displayed.
`blanking`	Points that are indicated as being hole points in the grid file are not listed.
`noblanking`	Blanking data in the grid file is not examined to see if a point should not be listed.
`raw`	Outputs data without headers or form feeds in a wide 132 column format.
`nowrap|wrap` *linelength*	Specfies whether or not output lines are to be wrapped. The default is `wrap`, with a *linelength* of 132 characters if `raw` is specified, and 80 characters otherwise.

The `list` command lists the values of all of the selected variables in all of the selected subsets. The command can be used to find the minimum and maximum value of the selected variables and their location. It can also be used to find values that are inside or outside a specified range. The `raw` and `nowrap` formats are useful for writing data for programs to read.

For subsets of unstructured grids selected with the `surface` command, all points associated with the selected surface are listed.

Example 1

```
zone 1
   subset i all j 1 k 15
variable x inches; p pinf; T Tinf
list
```

Example 2

```
! Find all bad points in a solution
variable p pinf
list outside range 0.0 to 100.0
```

See Also: The `subset` command for how to define the domain to be listed; the `variable` command for how to select variables and their units.

`neumcp` — Generate a NEUMCP file

```
neumcp output file
```

`output` *file* Specifies the name of the file to receive the data. A file extension of "*.neu*" will be provided if none was supplied. The file contains only text data and can be freely interchanged between computer systems.

The `neumcp` command produces a NEUMCP file that is used by several organizations within Boeing.

`surface` commands may be used to define the surfaces or `subset` commands may be used to specify surfaces or groups of surfaces. If a `subset` command is used to define a surface, it should define the surface so that the normal vector would point towards the viewer if viewed from the "front" or "outside" of the surface.

If a `variable` command is not specified, a standard NEUMCP file will be produced which contains the geometry and pressure coefficient. The units of the geometry information will be the default length unit. If a `variable` command is specified then the geometry and the selected variables will be written to the file. Such a file will probably be usable only by a custom program.

Note: This command is not available for unstructured grids.

Example

```
units inches                 ! Specify the default length unit for geometry
zone 1
   surface j 1 i 9,last      ! Omit the singular axis
zone 2
   surface j 1
neumcp output tma2
```

See Also: The `surface` command for how to define surfaces; the `subset` command on how to define surfaces and normal vectors; the `units` command on how to define the default length unit; the `variable` command on how to select variables and their units.

`plot3d` — Generate files for PLOT3D

```
plot3d [xyz xyzfile] [q qfile] [function functionfile] [names namesfile] -
   [2D|3D] [mgrid] [blank] [formatted|unformatted|iris]
```

`xyz` *xyzfile*	Specifies the name of the PLOT3D grid file.
`q` *qfile*	Specifies the name of the PLOT3D solution file.
`function` *functionfile*	Specifies the name of the PLOT3D function file.
`names` *namesfile*	Specifies the name of the PLOT3D function names file.
`2D`	Indicates that data is to be written so that it can be read using the `READ/2D` command.
`3D`	Indicates that data is to be written so that it can be read using the `READ/3D` command.
`mgrid`	Indicates that data is to be written so that it can be read using the `READ/MGRID` command. This option is automatically implied if more than one subset is specified. If only one subset has been specified, then this option must be specified if multigrid format is desired.
`blank`	Indicates that IBLANK data is to be included in the xyz file. The file must be read with the `READ/BLANK` command.
`formatted`	Indicates that data is to be written so that it can be read with the `READ/FORMATTED` command. Such files can be freely exchanged between computer systems. This is the default option.
`unformatted`	Indicates the data is to be written so that in can be read with the `READ/UNFORMATTED` command. Because these files contain binary information, they can be interchanged only between like computer systems.
`iris`	Indicates the data is to be written so that it can be read with the standard binary `READ` command on a Silicon Graphics computer system (or any IEEE Big Endian system like HP, Sun, or PC). This option is *not* available on Cray computers.

This command produces files for the program PLOT3D or any other program that accepts files in PLOT3D format. Currently the output is always single precision.

The data written to the xyz file will be written in units of default length units, defined using the `units` command.

If a `variable` command has not been specified, the standard five variables (four if 2D) will be written to the q or function file with the standard PLOT3D normalization. (I.e., the variables written are static density, momentum in the Cartesian coordinate directions, and total energy per unit volume, with density non-dimensionalized by the freestream static density ρ_∞, velocity non-dimensionalized by the freestream speed of sound a_∞, and total energy per unit volume non-dimensionalized by $\rho_\infty a_\infty^2$.)

If a `variable` command is specified then *exactly* five variables (four if 2D) must be selected for output to the q file. If the user wants to plot fewer than five variables, the remaining variables must be filled with any valid variable. If using PLOT3D the resultant file must be read with the `/NOCHECK` qualifier. The first variable is plotted as `FUNCTION` 100, second as 160, third as 161, fourth as 162 (if not 2D) and the last as 163.

Alternatively, an *arbitrary* number of variables may be selected for output to a PLOT3D function file. When reading function files with PLOT3D, be sure to use the `READ/NOCHECK` command to avoid warning messages. Note that some software products, such as Tecplot, will only read function files if the file structure (i.e., 2D, mgrid, formatted, etc.) is described manually and the solution file style is set to Plot3d Function rather than Plot3d Classic (i.e., q files). The function names file is a simple text file used by some commercial software products (i.e., Fieldview, Tecplot) to identify the names of the variables stored in the PLOT3D function file.

Note: This command is not available for unstructured grids.

Example 1

Create standard PLOT3D "xyz" and "q" files for all of the solid body surfaces.

```
units inches                     ! The default length unit for the "xyz" file
zone 1
   subset i 9,last j 1 k all     ! Omit the singular axis
zone 2
   surface j 1
plot3d xyz test.xyz q test.q unformatted
```

Example 2

Create a non-standard 3D "q" file for plotting two chemical species mass fractions. Note that three other variables must be specifed just to fill up five slots.

```
units inches
subset i all j all k all    ! Define a default subset
zone 1
zone 2
variable H2O; CO2; u; u; u
plot3d q species.q unformatted
```

The resultant file must be read with the PLOT3D `READ/NOCHECK` qualifier. *H2O* will be plotted with function 100 and *CO2* with function 160.

Example 3

Create a 3D "function" file and function "names" file for plotting two chemical species mass fractions. Note that this function file will only contain two variables.

```
units inches
subset i all j all k all    ! Define a default subset
zone 1
zone 2
variable H2O; CO2
plot3d function species.fun names species.nam unformatted
```

The resultant file must be read with the PLOT3D `READ/NOCHECK` qualifier. *H2O* will be plotted with function 100 and *CO2* with function 160.

Example 4

Create a 3D "function" file and function "names" file for plotting more than five variables. Note that such files can become quite large, depending on the number and size of subsets selected.

```
subset i all j all k all                   ! Define a default subset
zone 1 to last
units inches                               ! The default length unit for the ''xyz'' file
plot3d xyz plot3d.xyz unformatted
units fps                                  ! The default length unit for the ''function'' file
variable rho; u; v; w; p lbf inch; T; M
plot3d function plot3d.fun names plot3d.nam unformatted
```

See Also: The `subset` command for how to define the domain; the `units` command for how to define the default length unit; the `variable` command for how to select variables and their units.

`smooth` — Smooth a polar rake *.cgf* file

```
smooth [rake] rings num_extra_rings legs num_extra_legs
```

`rings` *num_extra_rings*	Specifies the number of rings to add between existing rings.
`legs` *num_extra_legs*	Specifies the number of legs to add between existing legs.

The `smooth` command creates new rings and legs between existing rings and legs using radial and linear interpolation. The user specifies the number of rings and legs to add between the current rings and legs, all original points are maintained. The rake must be in the same order that the `rake polar` command generates, that is I running from the inner ring to the outer ring and J running around the legs.

Note: The new grid and solution is saved over the old grid and so no new file is generated! This command only works on polar 2D structured grids.

Example

```
grid polar.cgf
solution polar.cgf
zone 1
smooth rake rings 3 legs 5
unit inches
cut at z 0.0
plot color contours
```

See Also: The `subset` command for how to define the domain; the `units` command for how to define the default length unit; the `cut` command for how to set cutting planes; the `plot` command for how to plot color contours.

7 Plotting Commands

`plot` — Plot *x*-*y* data

```
plot [data plotfile] [parameter parmfile] [merge plotfile [merge plotfile ...]]  -
    [begin firstplot] [end lastplot] [{exchange|switch} [coordinates]] -
    [autoscale|noautoscale]
```

`data` *plotfile*	If no subsets or variables are selected, *plotfile* is the name of a file to be plotted. This file may have been created by a previous execution of the `genplot` command or any other program capable of generating a file of the proper type. If this parameter is not specified then a prompt will be issued to ask for the name of the file to plot. A file extension of '*.gen*' is assumed if none is specified. If this parameter is not specified, the `genplot` command will be internally invoked to generate a temporary file called *genplot.tmp*. This file will then be plotted.
`parameter` *parmfile*	Specifies the name of a parameter file that will be used to set the various attributes of the plot. This file will generally be created during a previous execution of `plot`. If this parameter is not specified then the default parameters will be used.
`merge` *plotfile* `merge` *plotfile* ...	Specifies the names of one or more files to be plotted simultaneously with *plotfile*. The first plot of each file will be plotted together on one plot, followed in a similar manner for the second and subsequent plots in each file.
`begin` *firstplot*	Specifies the number of the plot within the plot files at which plotting is to begin. This parameter is used to skip the viewing of plots at the beginning of a file. If this parameter is omitted, plotting begins with the first plot in the files.
`end` *lastplot*	Specifies the number of the plot within the plot files at which plotting is to stop. The parameter is used to stop plotting before all plots in the files have been viewed. If this parameter is omitted, plotting will stop after the last plot in the files.
`exchange|switch [coordinates]`	Indicates the *x* and *y* coordinates of the data should be exchanged before plotting.
`autoscale|noautoscale`	`autoscale` causes the plotting scale to be reset when a new plot is selected, `noautoscale` keeps the previous scale.

The `plot` command is used to produce *x*-*y* plots from existing GENPLOT files (produced externally or with the `genplot` command) or from currently selected subsets and variables.

The appearance of the plot can be controlled during the plotting process. Axis type (linear or logarithmic), scaling, grid overlays, line and symbol types and titles are some of the items that can be changed. The parameters that control appearance can be saved into a parameter file for use on subsequent `plot` commands. Hardcopy may be produced in either Boeing intermediate plot file (IPF) or PostScript format.

The `merge` parameter is useful for producing comparison plots between different solutions or between a solution and experimental data. Experimental data can be cast into GENPLOT format following the layout in Appendix A.

Example: Plot a single file

```
plot data pltfile                    ! Plot the whole file
plot data pltfile begin 2 end 2      ! Plot only the second plot
```

Example: Plot multiple files

```
plot data expdata merge soln1 merge soln2
```

Example: Plot data from selected subsets

```
zone 1
   surface j 1      ! Define the surface to be cut
units inches        ! Specify default length unit
cut at z 150.0      ! Constant z cutting plane
variable x inches; Cp
plot                ! Plot it up ...
```

See Also: The `genplot` command for how to generate plot files; the `set plot` command for how to set plotting parameters; Appendix A for a description of the GENPLOT file format.

`plot contours` — Produce contour plots

```
plot [color|grey|gray] {contours|grid} [edge|alternate] [shade num] -
   [data plotfile] [inverted] [parameter parmfile] [begin firstplot] [end lastplot] -
   [variable varname] [{exchange|switch} [coordinates]] [autoscale|noautoscale]
```

`color`	A filled color contour plot will be generated rather than a line contour plot.
`contours`	Color contours are plotted in either line or shaded mode as selected.
`grid`	Plots the underlying unstructured grid surface.
`grey|gray`	A filled grey scale contour plot will be generated rather than a line contour plot.
`edge`	Filled contours will be outlined with solid lines to highlight changes in contour levels.
`alternate`	Like `edge` except every other line is a dashed line rather than a solid line.
`shade` *num*	Specifies the number of colors or shades of grey to use for constructing the shading ramp. If `shade` is not specified, one color or grey shade will be used for each contour level. Increasing the number of shades gives a better approximation to continous shading at the expense of increased drawing time or plot file sizes. Exceeding 64 is rarely beneficial for hardcopy devices.
`inverted`	Inverts the color map used in color contour plotting.
`data` *plotfile*	Same as the `plot` command except the default file extension is *.gpc*.
`parameter` *parmfile*	Same as the `plot` command.
`begin` *firstplot*	Same as the `plot` command.
`end` *lastplot*	Same as the `plot` command.
`variable` *varname*	Plots the specified variable and exits to next plot. *varname* can be any valid variable with scaling. If whitespace is in the name or scaling then it must be put in double quotes.
`exchange|switch [coordinates]`	Same as the `plot` command.
`autoscale|noautoscale`	Same as the `plot` command.

The `plot contours` command is used to produce contour plots from files generated with the `genplot surface` command. The contour levels to be plotted are determined by the `contour` command. If a `contour` command is not specified `contour automatic levels` *15* will be assumed.

The display attributes of the contour levels and walls are controlled by the `set plot curve` command or controlled during the plotting process. The attributes for the first curve determine the appearance of the walls. For line plots, the attributes of the second curve determine the appearance of intermediate (nonannotated) contour levels and the attributes of the third and higher curves determine the attributes of the major (annotated) contour levels. For filled plots that are edged, the attributes of curve two determine the appearance of the edges. For filled plots with alternating edging, the attributes of curves two and three determine the appearance of the alternating edges.

For filled contour plots, the `contour` command determines the labels printed in the legend and where contour edge lines appear. Do not use more contour levels to achieve more continous shading; use the `shade` parameter to increase the number of shades. Also note that for filled plots, the contour levels indicated by the labels in the legend are at the center point of the contour bands. The edge lines are at the midpoint *between* contour levels.

The plotting interface allows selection from a menu of the cut to plot, so that cuts can be picked in any order and you can go back and forth between cuts. For each cut you can derive variables from existing ones in the genplot common file as long as the variable derivation does not need the grid (like `y+`). For vectors you can select from available or derivable vector fields and change the vector scale. When you are deriving a variable it is specified with the same syntax as the `variable` command. Thus unit overrides like `pinf`, `ft`, etc may be used. The grid units of the plot are the units set when the `plot contours` command was invoked. If no units were specified the grid will have the default units of feet.

Example

Plot a file previously generated with the `genplot surface` command.

```
plot contours data pltfile
```

Example

Generate filled color contour plots of total pressure recovery and Mach number at the engine face. In this case the file *pltfile* is generated from the currently selected subsets and variables and immediately plotted. *pltfile* is retained for future plotting.

```
zone 1
   subset i all j all k all     ! Define the volume to be cut
units inches                    ! Specify default length unit
cut at x 150.0                  ! Constant x cutting plane
variable p0 p0inf; M
genplot surface output pltfile
plot color contours edge
```

See Also: The `genplot surface` command for how to generate files for contour plotting; the `plot` command for a more detailed description of the parameters; the `contour` command for how to set the contour levels; the `set plot` command for how to set plot parameters; the `variable` command for specifying variables.

`contour` — Specify contour levels for contour plots

```
contour automatic [levels levels] |
contour increment increment |
contour manual range [; range ...]
```

The `contour` command determines the contour levels for the `plot contours` command.

For filled contour plots, each contour level will be assigned a label in the legend and lines will optionally be drawn at the midpoints *between* contour levels. Conceptually, each contour band is centered about a specified contour level and has a width equal to the contour increment.

There are three possible formats for the `contour` command, as follows:

```
contour automatic [levels levels]
```

levels	Specifies the maximum number of contour levels that will be generated.

The `contour automatic` command indicates that the contour levels will be determined automatically from the minima and the maxima of the data to be plotted. No more than *levels* levels (default 15) will be generated.

```
contour increment [increment]
```

increment	Specifies the increment between adjacent contour levels. The number of contour levels will be set so as to enclose the extremes of the data being plotted.

```
contour manual range [; range ...]
```

range	Specifies the contour levels to be plotted, in the form *start*`[`,*end*,*incr*`]`.

`plot vectors` — Produce vector plots

```
plot vectors [data plotfile] [scale scale] [parameter parmfile] [begin firstplot] -
   [end lastplot] [variable varname] [{exchange|switch} [coordinates]] -
   [autoscale|noautoscale]
```

`data` *plotfile*	Same as the `plot` command except the default file extension is *.gpc*.
`scale` *scale*	A scaling factor for the vector length. The magnitude of a vector is multiplied by *scale* and the result is the length of the vector in engineering units that will be plotted. The default value for *scale* is 1.0.
`parameter` *parmfile*	Same as the `plot` command.
`begin` *firstplot*	Same as the `plot` command.
`end` *lastplot*	Same as the `plot` command.
`exchange|switch [coordinates]`	Same as the `plot` command.
`autoscale|noautoscale`	Same as the `plot` command.
`variable` *varname*	Plots the specified vector variable and exits to next plot. *varname* can be any valid vector variable with scaling. If whitespace is in the name or scaling then it must be put in double quotes.

The `plot vectors` command generates vector plots.

See Also: The `genplot surface` command for how to generate files for vector plotting; the `plot` command for a more detailed description of the parameters; the `set plot` comand for how to set plot parameters.

`set plot` — Set plot parameters

The `set plot` command allows the user to modify the appearance of subsequent plots. The values specified by the command override any values derived from the plot files or parameter files. Any value may be specified as `default` which causes the value to assume its default value or be gotten from a plot file or parameter file.

```
set plot default
```

Sets all user-definable plot parameters to their default values.

```
set plot device [ps|ipf] [-color] [-f filename] [-h height_in_inches] -
                [-w width_in_inches] [-o portrait|landscape]
```

Sets the current device on which plots are to appear, and some attributes of that device. Specifying a device and its associated parameters causes the selected device to be used until another `set plot device` command is issued.

`ps`	Create a PostScript file. This is the default device.
`ipf`	Create a Boeing intermediate plot file (IPF), a file format used internally at Boeing.
`-color`	Create a color plot. The default is black and white.
`-f` *filename*	The name of the output file. The default file names are *plotfile.ps* or *plotfile.ipf*, for a PostScript or Boeing intermediate plot file, respectively.
`-h` *height_in_inches*	The plot area height in inches. The default is 7.5.
`-w` *width_in_inches*	The plot area width in inches. The default is 10.0.
`-o portrait|landscape`	The plot orientation. The default is `landscape`.

```
set plot {width|height} value
```

Sets the width and height of the total plotting area in inches. This must be sufficient to include the plot, annotations, titles and legends. The default values are a width of 10.0 and a height of 7.5 inches. This option is not currently used and the values should not be altered.

```
set plot symbol height value
```

Sets the height in inches of the symbols used as point markers in the plot. The default value is 0.10.

```
set plot {x|y} origin value
```

Sets the x or y location in inches of the lower left corner of the plot region. For x-y and vector plots the defaults are (1.5,1.0). For contour plots the defaults are (1.25,0.75).

```
set plot {x|y} page [size] value
```

This is a synonym for `set plot width` (`x`) and `set plot height` (`y`).

```
set plot background -
   {default|black|white|red|green|yellow|blue|magenta|cyan}
```

Sets the background color for the plot. If the background color is black, the foreground color, used for axes, labels, etc., will be white. For all other background colors, the foreground color will be black. The default background color is black.

```
set plot curve curve# -
   line color {foreground|black|white|red|green|yellow|blue|magenta|cyan}
```

Sets the color for the selected curve. The default for all curves is `foreground` (i.e., white if the background color is black, and black otherwise).

```
set plot curve curve# line style {none|solid|dotted|dot-dash|dash|long-dash}
```

Sets the line style for the selected curve. For *x*-*y* plots the defaults for the first five curves are `solid`, `dotted`, `dot-dash`, `dash`, and `long-dash` respectively, with the remainder being `solid`. For contour plotting, the first curve defines the appearance of the walls, and defaults to `solid`. The second curve defines the appearance of intermediate (unannotated) contour levels and defaults to `dotted`. Curves three and up define the appearance of the major (annotated) contour levels

```
set plot curve curve# symbol type {none|square|octagon|triangle|plus| -
   x|diamond|up-arrow|x-superbar|z|y|x-square|star|hourglass|bar}
```

Sets the symbol to appear at data points on an *x*-*y* plot or as level indicators on a contour plot.

```
set plot curve curve# title "title"
```

Sets the title to appear in the legend for the specified curve. The title will be truncated to either 64 or 32 characters, depending on whether there are one or two columns in the legend.

```
set plot {x|y} axis annotation {positive|negative}
```

Establishes on which side of the axis the title and annotations are to appear. `positive` indicates on the counter-clockwise side, `negative` on the clockwise side. The default value for the *x* axis is `negative`, for the *y* axis it is `positive`.

```
set plot {x|y} axis annotation {horizontal|vertical|mixed}
```

Sets the orientation of the annotation and titles relative to the axis. `horizontal` indicates the title and annotations are parallel to the axis, `vertical` indicates the items are perpendicular to the axis, and `mixed` indicates the annotations are horizontal and the titles are vertical. The defaults are `horizontal` for the *x* axis and `vertical` for the *y* axis.

```
set plot {x|y} axis direction {increasing|decreasing}
```

Indicates whether values are to increase or decrease as they go from left-to-right or top-to-bottom.

```
set plot {x|y} axis grid {none|single|half|quarter|fifth|tenth}
```

Specifies the type of gridding to be applied to the plot.

```
set plot {x|y} axis increment value
```

Specifies the increment (in engineering units) between major tic marks on an axis. By default, the increment is automatically determined from the minimum and maximum values (either determined from the data to be plotted or specified by the user). If *value* is positive, the minimum and maximum values are rounded down and up respectively to the nearest multiple of the specified increment. If *value* is negative, the minimum value and the absolute value of the increment are used verbatim and the maximum value is determined by the length of the axis.

```
set plot {x|y} axis length value
```

Sets the length of the specified axis in inches. If the length is specified as a negative value and automatic scaling is used, the major tic marks will be forced to be on one inch physical increments. This is used if a resultant hardcopy plot file will have a grid overlay applied. For x-y plots and vector plots, the default lengths are 8.0 inches for the x axis and 5.0 inches for the y axis. For contour plots, the defaults lengths are 7.0 inches for the x axis and 5.25 inches for the y axis.

```
set plot {x|y} axis location {{left|bottom}|{right|top}|zero}
```

Specifies the location of the axis on the plot. The default for x axis is `bottom` and for y axis it is `left`.

```
set plot {x|y} axis matching {on|off}
```

Specifies whether the scale factor for one axis should be copied from the other axis. This option is used if both axes must have identical scaling. If both axes have this flag set, the smallest of the two scale factors will be used. For x-y plots the default is `off` for both axes. For contour and vector plots the default value is `on` for both axes.

```
set plot {x|y} axis {minimum|maximum} value
```

Specifies the minimum or maximum value (in engineering units) to be plotted on the specified axis. If one of the values (minimum or maximum) is specified, then both must be specified. If neither value is specified or they are both zero, the values are determined from the data to be plotted. The minimum and maximum values are only suggestions. These values will be rounded down or up appropriately to an integral multiple of the increment. See the `set plot increment` command for more details.

```
set plot {x|y} axis type {linear|logarithmic}
```

Defines the type of scaling for the specified axis. The default value for both axes is `linear`.

```
set plot [plot#] {x|y} axis title “title”
```

Sets the title for the specified axis. The default for *plot#* is one, and for *title* is the one read from the plot file.

```
set plot [plot#] {primary|secondary} title “title”
```

Sets the primary or secondary title. The default for *plot#* is one, and for *title* is the one read from the plot file.

```
set plot security {upper|lower} title “title”
```

Sets one of the security titles that are plotted in the lower left or upper right hand corners of the plot. The titles are printed in a heavy Roman red font, are limited to 77 characters, and are above/below all user definable titles.

```
set plot date {on|off}
```

Indicates whether the current date is to be displayed in the lower left corner of the plot. If both the date and time are to be displayed, the time will follow the date.

```
set plot time {on|off}
```

Specifies whether the current time is to be displayed in the lower left corner of the plot. If both the date and time are to be displayed, the time will follow the date.

`set width` — Set width or coordinate parameters

```
set {xwidth | ywidth | zwidth | zcoordinate}
```

`xwidth`	Sets the width (constant) coordinate to x for 2D (`kdim` = 1) grids.
`ywidth`	Sets the width (constant) coordinate to y for 2D (`kdim` = 1) grids.
`zwidth`	Sets the width (constant) coordinate to z for 2D (`kdim` = 1) grids.
`zcoordinate`	Sets the z coordinate to be used as a real coordinate not as a width field. This is used when integrating 3D surfaces stored as 2D (`kdim` = 1) grids.

The `set width` command is used to tell CFPOST how to treat a 2D grid for integrations and cutting planes. If one of the width fields is selected then a quasi-3D grid is generated by reflecting the selected coordinate about zero with a magnitude of one half the coordinate value. If `zcoordinate` is set then no quasi-3D grid is generated and the coordinates are just read out of the file without modification.

Example 1

Have 3D surface grids copied out of the full zone and integrate pressure.

```
grid surfs.cgd
solution surfs.cfl
zone 1 to last
set zcoordinate
integrate force output force.lis
```

Example 2

We wish to cut a polar rake file which is in the x constant plane.

```
grid polar.cgf
solution polar.cgf
unit inches
zone 1
set xwidth
cut at x 0.0
plot color contour
```

See Also: The `plot` command for a more detailed description of the parameters; the `contour` command for how to set the contour levels; the `set plot` command for how to set plot parameters; the `cut` command on how to set cutting planes; the `integrate force` command for force integrations.

`visual3` — Initialize the interactive 3D/2D VISUAL3 environment

```
visual3 initialize [colormap file] [mirror x|y|z] [blank|noblank] -
   [big3dw|big2dw] [no3dsurf] [multizone] [grayscale]
```

`colormap` *file*	Specifies a user defined color map file to be used in color contour plotting.
`mirror x|y|z`	Enables mirroring about the selected axis.
`blank|noblank`	Flag to use/ignore iblanking data in the grid file (not used for particle tracing).
`big3dw|big2dw`	Makes either the 3D or 2D window the largest window.
`no3dsurf`	Causes no surfaces to be displayed in the 3D subsets. This is useful when you have all the surfaces selected and a few full 3D zones in which to perform iso-surface, cutting plane, or particle trace displays, since duplicate or unwanted surfaces will not come up displayed.
`multizone`	Allows streamline traces to go between zones. This adds significant overhead to the traces in VISUAL3. The default is single zone.
`grayscale`	Displayed surfaces come up in gray scale instead of in color contours.

The 3D/2D visualizer VISUAL3 developed at MIT is now available in CFPOST. CFPOST provides an interface to the data used in the visualization, while VISUAL3 itself provides full 3D imaging with real time rotations/translations, user selectable cutting planes which are drawn in a separate 2D window, and particle/ribbon tracing.

The CFPOST Interface

In CFPOST you select the subsets and the variables (16 max) that you wish to view. The variables selected are programmed onto the keys `a`–`q` for selection (see "The VISUAL3 Environment" below). Unit overrides such as `pinf` are used in variable display. If you are displaying surfaces you should use the CFPOST `surface` command to ensure proper ordering of the indices. Currently only structured grids are supported. Once you set up the subsets and variables VISUAL3 is started using this command.

The VISUAL3 Environment

When VISUAL3 is initialized in a CFPOST session a number of windows appear on your screen including the 3D/2D plotting windows, a dials/groups window, a key/function window, and an auxiliary window. Each window has a specific function which may be invoked by pressing a key when that window is active!! The question mark may be entered in any window to give a help list of functions, and their associated keys, that are available in that window. Typed input and program messages appear in the original CFPOST window. The tilde may also be entered in any window to create a bitmap image of that window. A utility program exists to convert the bitmap image files into PostScript or TIF formats. Below is a brief description of each window's main function.

3D/2D Window

The 3D window is the drawing area for the 3D surface/volume plots and the 2D window displays the current cutting plane information if cutting planes are active. Some of the main functions in the 3D window include selection of the current contour variable through the press of the `a`–`q` keys (see "The CFPOST Interface" above), the selection of user defined cutting planes (`F4`), center/scale (`+`), isosurfaces (`F7`), streamlines/ribbons/tubes mode toggle (`|`), and program termination (`Esc`). In the 2D window the mouse buttons may be used to place starting points for streamline traces going in the upstream (left button), both (middle button), and downstream (right button) directions. Several new custom toggles which have been added are: `B` (set black background), `W`-set white background), `X|Y|Z` (x, y, or z mirror toggle), and `E` (edge outline toggle).

Dials/Group Window

This window has two main modes: a Dial Box mode and a Surface Group Display mode. The `s` key is used to toggle between the two modes.

The Dial Box mode is used if you don't have a dial box attached to the system. It has the usual translate/rotate/zoom areas as well as a cutting plane scan area. Each area is defined by two concentric circles, the inner circle is for fast motion and the outer circle is for regular motion. To activate the area place the cursor in the desired circle and press and hold the right/left mouse button to move right/left, up/down, etc. If you press the center mouse button while not in any of the circular areas you toggle between the 3D/2D window/Key dials.

The Surface Group Display modes lets you turn on and off the currently defined groups. Since VISUAL3 doesn't know about zones and I/J/K planes you must define a named group which is a list of all cells which make up that group. Currently CFPOST defines the six faces of each zone as a group with the label "ZONE *nn face*" where *face* is one of `I1`, `IMAX`, `J1`, `JMAX`, `K1`, and `KMAX`. By default the `K1` face of each zone is automatically displayed. Thus since the CFPOST `surface` command creates `K1` surfaces, they will automatically displayed upon initialization. For each Surface Group there are four different display options available in the four boxes to the right side of the name. The first box enables color rendering in either solid or transparent mode. If the box is white it is in solid mode, if it is grey it is in transparent mode, and if it is black it is disabled. The second box enables solid shading of the surface, other boxes enable thresholding based on the current threshold function (see "Key/Function Window" below).

Key/Function Window

This window displays the current function color bar as well as the current threshold function and range. The main functions in this window include query (`q`) which gives the current function range, function limits (`f`) which sets the current contour range, threshold range (`t`) which sets the threshold limits on the display if that surface has thresholding enabled (see "Dials/Group Window" above), and set isosurface value (`z`).

Miscellaneous

When you startup VISUAL3 it reads/creates two files, *.Visual3.struc* containing connectivity data for particle tracing and *.Visual3.setup* containing default/saved views and other miscellaneous information. If something doesn't come up scaled correctly or if particle traces are not working then you may need to get out of the program and delete these files.

Limitations

VISUAL3, like PLOT3D and FAST, uses in-core memory so be aware!! VISUAL3 runs on the SGI's and on HP's with the PEX graphics library installed. PEX comes free with HP operating systems 9.05 and greater. You cannot run VISUAL3 as an X-terminal, you must be on the workstation console.

Miscellaneous Setup

In order for VISUAL3 to look right, you need to make the following additions to your *.Xdefaults* file.

```
!
! VISUAL2/3 fonts
```

```
!
Visual*small:  6x10
Visual*medium: 8x13
Visual*med2:   -adobe-courier-bold-r-normal--17-120-100-100-m-100-iso8859-1
Visual*large:  -adobe-courier-bold-r-normal--25-180-100-100-m-150-iso8859-1
```

For VISUAL3 to work well you should set your X-Windows focus policy to "pointer". To accomplish this the following line should be uncommented in your *.Xdefaults* file.

```
Mwm*keyboardFocusPolicy: pointer
```

Appendix A. GENPLOT File Format

CFPOST's `plot` command may be used to produce x-y plots from information stored in a GENPLOT file. GENPLOT files may be created by CFPOST's `genplot` command, or using any appropriate ASCII text editor.

There are two formats allowed for GENPLOT files. The first contains multiple curves on each plot, with each curve defined by its own set of x-y coordinate pairs. The second format also contains multiple curves on a plot, but with the same set of x coordinate values for each curve.

The curve titles specified in the file will automatically be truncated to either 64 or 32 characters, depending on whether there are one or two columns in the legend.

Comments may be included in the file by using a # as the first character in the line (for CFPOST 3.173 and later).

<u>*Format 1*</u>: Multiple x *vs* y curves (produced by `genplot multiple plots`)

Main plot title		Line 1
Plot 1 sub-title		Line 2
x-axis title		Line 3
y-axis title		Line 4
# of curves		Line 5
Curve 1 title		Line 6
# of points in curve 1		Line 7
$x1_1$, $y1_1$		Line 8
...		
$x1_n$, $y1_n$		Line $8 + n - 1$
Curve 2 title		Line $8 + n$
# of points in curve 2		Line $8 + n + 1$
$x2_1$, $y2_1$		Line $8 + n + 1 + 1$
...		
$x2_m$, $y2_m$	(End of last curve in first plot)	Line $8 + n + 1 + m$
...	(More plots)	

There may be multiple plots in one plot file. The first line is the main title for all plots in the file. Lines 2 through m represent a plot. A plot has one or more curves as described by lines 6 through $8 + n - 1$. Each curve in a plot may be defined by a different number of points.

Format 2: Multiple y *vs* single x (produced by `genplot multiple variables`)

Main plot title	Line 1
Plot 1 sub-title	Line 2
x-axis title	Line 3
y-axis title	Line 4
Negative # of curves	Line 5
Curve 1 title	Line 6
Curve 2 title	Line 7
...	
Curve n title	Line $6+n-1$
# of points in curves	Line $6+n$
$x1_1$, $y1_1$, $y1_2$, ..., $y1_n$	Line $6+n+1$
...	
xm_1, ym_1, ym_2, ..., ym_n	Line $6+n+m$

Appendix B. `rake file` Format

The file format used by the `rake file` command is a form of the EAGLE ASCII surface and volume grid file. The program will accept EAGLE surface and grid files as well as less rigidly formatted files of the same type. Extensions have been made to allow the input of curves as well as surfaces and volumes. A file may contain a mix of curves, surfaces and volumes, and the file format should be based on the highest order surface in the file.

Following are some FORTRAN code fragments that demonstrate the method for creating a file of the proper format. If a file is going to contain a mix of curves, surfaces and volumes, the number of zones and their dimensions is written only at the start of the file with the information for all curves, surfaces and volumes.

Writing a 1-D File (Curves)

```
      Real x(ni,nz),y(ni,nz),z(ni,nz)   ! The coordinate data
      Integer imax(nz)                  ! The dimensions

      Write (11,*) nz                   ! Number of zones
      Do 100 n = 1, nz                  ! Dimensions
         Write (11,*) n,imax(n)
100   Continue
      Do 210 n = 1,nz
         Do 200 i = 1,imax(n)
            Write (11,*) x(i,n),y(i,n),z(i,n)
200      Continue
210   Continue
```

Writing a 2-D File (Surfaces)

```
      Real x(ni,nj,nz),y(ni,nj,nz),z(ni,nj,nz)
      Integer imax(nz),jmax(nz)

      Write (11,*) nz
      Do 100 n = 1, nz
         Write (11,*) n,imax(n),jmax(n)
100   Continue
      Do 220 n = 1,nz
         Do 210 j = 1,jmax(n)
            Do 200 i = 1,imax(n)
               Write (11,*) x(i,j,n),y(i,j,n),z(i,j,n)
200         Continue
210      Continue
220   Continue
```

Writing a 3-D File (Volumes)

```
      Real x(ni,nj,nk,nz),y(ni,nj,nk,nz),z(ni,nj,nk,nz)
      Integer imax(nz),jmax(nz),kmax(nz)

      Write (11,*) nz
      Do 100 n = 1, nz
         Write (11,*) imax(n),jmax(n),kmax(n)
100   Continue
```

```
      Do 230 n = 1,nz
         Do 220 k = 1,kmax(n)
            Do 210 j = 1,jmax(n)
               Do 200 i = 1,imax(n)
                  Write (11,*) x(i,j,k,n),y(i,j,k,n),z(i,j,k,n)
200            Continue
210         Continue
220      Continue
230   Continue
```

Appendix C. Equations Used by CFPOST

C.1 Variable Derivations

The following equations are used by CFPOST for deriving a requested variable that is not present in the solution file. If a variable cannot be derived by one of the following equations or by the simple multiplication or division by ρ, then the requested variable must exist within the solution file. Note that multiple derivations may be used to derive a variable. For example, when deriving the total temperature T_0, the temperature T and Mach number M may themselves be derived.

The derivation of variables that include the gas constant R or the ratio of specific heats γ will take real gas effects into account if sufficient information is available. For frozen and finite-rate chemistry flows, species properties are read from a chemistry data (*.chm*) file specified with the `chemistry` command. Otherwise, real gas effects are modeled by multiplying R_∞ by the compressibility factor Z, and using the effective specific heat ratio β in place of γ.

Note that some variables require the specification of a surface, or normal direction into the flow field, and others require knowledge of “up” and “down” axes orientation. Variables denoted with (s) indicate that the surface and normal direction into the flow field must be specified properly using the `subset` or `surface` command. Variables denoted with (o) are affected by the axes orientation set by the `orientation` command.

Several variables may be computed in multiple ways, depending on the information available in the solution file. The equations are shown below in the order that CFPOST attempts to use them. If any of the needed quantities are missing from the solution file and cannot be derived, CFPOST will try using the subsequent equation, etc. Note though, that for frozen and finite-rate chemistry flows, if the necessary information isn't available CFPOST will abort with an error message, rather than use the perfect gas equations.

`rho` — *Density*

$$\begin{aligned}\rho &= \frac{p}{R_\infty T Z}\\ &= \frac{p}{R_\infty T}\end{aligned}$$

Z contains the local real gas effects, R/R_∞.

`rho0` — *Stagnation density*

For frozen and finite-rate chemistry,

$$\rho_0 = \frac{p_0}{R_\infty T_0 Z}$$

For a perfect gas,

$$\rho_0 = \rho\left(1 + \frac{\gamma_\infty - 1}{2}M^2\right)^{\frac{1}{\gamma_\infty - 1}}$$

`p` — *Pressure*

$$\begin{aligned} p &= \rho R_\infty T Z \\ &= \rho R_\infty T \\ &= (\beta - 1)\left(\rho e_0 - \rho\frac{V^2}{2} - \rho h_f\right) \\ &= (\gamma_\infty - 1)\left(\rho e_0 - \rho\frac{V^2}{2} - \rho h_f\right) \end{aligned}$$

Z contains the local real gas effects, R/R_∞.

`p0` — *Stagnation pressure*

For frozen and finite-rate chemistry,

$$p_0 = p\exp\left[\frac{1}{R_{mix}}\sum_{i=1}^{ns} C_i\frac{R_u}{MW_i}(\texttt{Sfun}_{i,T_0} - \texttt{Sfun}_{i,T})\right]$$

where `Sfun` is a function used to compute the entropy for a species as a function of temperature, from the curve fit data in the *.chm* file.

For a perfect gas,

$$p_0 = p\left(1 + \frac{\gamma_\infty - 1}{2}M^2\right)^{\frac{\gamma_\infty}{\gamma_\infty - 1}}$$

For a value in a rotating reference frame (i.e., `p0r`), the total temperature T_0 and Mach number M in the above equations are replaced by the corresponding values in the rotating reference frame.

`Cp` — *Pressure coefficient*

$$C_p = \frac{p - p_\infty}{q_\infty}$$

where $q_\infty = \rho_\infty(M_\infty a_\infty)^2/2$.

`deltap` — *Delta pressure*

$$\Delta p = p - p_\infty$$

`pp` — *Pitot pressure*

$$p_p = p\left(1 + \frac{\gamma_\infty - 1}{2}M^2\right)^{\frac{\gamma_\infty}{\gamma_\infty - 1}} \qquad \text{for } M \le 1$$

$$p_p = p\frac{\left(\frac{\gamma_\infty + 1}{2}M^2\right)^{\frac{\gamma_\infty}{\gamma_\infty - 1}}}{\left(\frac{2\gamma_\infty}{\gamma_\infty + 1}M^2 - \frac{\gamma_\infty - 1}{\gamma_\infty + 1}\right)^{\frac{1}{\gamma_\infty - 1}}} \qquad \text{for } M > 1$$

`Cpt` — *Total pressure coefficient*

$$C_{pt} = \frac{p_0 - p_\infty}{q_\infty}$$

where $q_\infty = \rho_\infty (M_\infty a_\infty)^2/2$.

`q` — *Dynamic pressure*

$$q = \frac{\rho V^2}{2}$$

`T` — *Temperature*

$$\begin{aligned} T &= \frac{p}{\rho R_\infty Z} \\ &= \frac{p}{\rho R_\infty} \end{aligned}$$

Z contains the local real gas effects, R/R_∞.

`T0` — *Stagnation temperature*

For frozen and finite-rate chemistry, iterate on T_0 until

$$\begin{aligned} &h_{mix,T_0} = h_{0,mix} \\ &\sum_{i=1}^{ns} C_i \frac{R_u}{MW_i} \texttt{Hfun}_{i,T_0} = \sum_{i=1}^{ns} C_i \frac{R_u}{MW_i} \texttt{Hfun}_{i,T} + \frac{V^2}{2} \end{aligned}$$

where `Hfun` is a function used to compute the enthalpy for a species as a function of temperature, from the curve fit data in the *.chm* file.

For a perfect gas,

$$T_0 = T\left(1 + \frac{\gamma_\infty - 1}{2} M^2\right)$$

`h` — *Sensible enthalpy*

$$\begin{aligned} h &= \beta e_i \\ &= \gamma_\infty e_i \end{aligned}$$

`h0` — *Sensible stagnation enthalpy*

$$\begin{aligned} h_0 &= \beta e_i + e_k \\ &= \gamma_\infty e_i + e_k \end{aligned}$$

`ei` — *Sensible internal energy per unit volume*

$$\begin{aligned} e_i &= \frac{p}{\rho(\beta - 1)} \\ &= \frac{p}{\rho(\gamma_\infty - 1)} \end{aligned}$$

`ek` — *Kinetic energy per unit volume*

$$e_k = \frac{V^2}{2}$$

`e0` — *Absolute stagnation energy per unit volume*

$$\begin{aligned} e_0 &= \frac{\rho e_0}{\rho} \\ &= \frac{1}{\rho}\left[\rho(e_0)_r + \frac{\rho}{2}(\boldsymbol{\omega}\times\mathbf{r})^2 - \frac{\rho}{2}(\mathbf{g}\cdot\mathbf{r})\right] \\ &= e_i + e_k + h_f \end{aligned}$$

`Ma` — *Equivalent isentropic Mach number*

$$\begin{aligned} M_a &= \sqrt{\frac{2}{\beta-1}\left[\left(1+\frac{\beta-1}{2}M_\infty^2\right)\left(1+\frac{\beta C_p}{2}M_\infty^2\right)^{\frac{1-\beta}{\beta}} - 1\right]} \\ &= \sqrt{\frac{2}{\gamma_\infty-1}\left[\left(1+\frac{\gamma_\infty-1}{2}M_\infty^2\right)\left(1+\frac{\gamma_\infty C_p}{2}M_\infty^2\right)^{\frac{1-\gamma_\infty}{\gamma_\infty}} - 1\right]} \end{aligned}$$

where C_p is the pressure coefficient.

`u` — *x velocity*

In a non-rotating reference frame,

$$\begin{aligned} u &= \frac{\rho u}{\rho} \\ &= u_r + (\boldsymbol{\omega}\times\mathbf{r})\cdot\boldsymbol{e_x} \\ &= 0 \end{aligned}$$

In a rotating reference frame,

$$\begin{aligned} u_r &= \frac{\rho u_r}{\rho} \\ &= 0 \end{aligned}$$

`v` — *y velocity*

In a non-rotating reference frame,

$$\begin{aligned} v &= \frac{\rho v}{\rho} \\ &= v_r + (\boldsymbol{\omega}\times\mathbf{r})\cdot\boldsymbol{e_y} \\ &= 0 \end{aligned}$$

In a rotating reference frame,

$$\begin{aligned} v_r &= \frac{\rho v_r}{\rho} \\ &= 0 \end{aligned}$$

`w` — *z velocity*

In a non-rotating reference frame,

$$\begin{aligned} w &= \frac{\rho w}{\rho} \\ &= w_r + (\boldsymbol{\omega} \times \mathbf{r}) \cdot \boldsymbol{e_z} \\ &= 0 \end{aligned}$$

In a rotating reference frame,

$$\begin{aligned} w_r &= \frac{\rho w_r}{\rho} \\ &= 0 \end{aligned}$$

<u>`V` — *Velocity magnitude*</u>

$$V = \sqrt{V^2}$$

For a value in a rotating reference frame (i.e., `Vr`), V^2 in the above equation is replaced by the corresponding value in the rotating reference frame.

The value of V^2 in a non-rotating reference frame is computed using one of the following equations. (As noted earlier, the equation used depends on the information available in the solution file. If any of the needed quantities in the first equation listed below are missing from the solution file and cannot be derived, CFPOST will try using the subsequent equation, etc.)

$$\begin{aligned} V^2 &= |\mathbf{V}^2| \\ &= u^2 + v^2 + w^2 \\ &= \frac{(\rho|\mathbf{V}|)^2}{\rho^2} = \frac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{\rho^2} \\ &= (|\mathbf{V}_{\boldsymbol{r}}| + |\boldsymbol{\omega} \times \mathbf{r}|)^2 \\ &= \frac{(\rho|\mathbf{V}_{\boldsymbol{r}}| + \rho|\boldsymbol{\omega} \times \mathbf{r}|)^2}{\rho^2} \end{aligned}$$

In a rotating reference frame, V_r^2 is computed using one of the following equations.

$$\begin{aligned} V_r^2 &= |\mathbf{V}_{\boldsymbol{r}}^2| \\ &= u_r^2 + v_r^2 + w_r^2 \\ &= \frac{(\rho|\mathbf{V}_{\boldsymbol{r}}|)^2}{\rho^2} = \frac{(\rho u_r)^2 + (\rho v_r)^2 + (\rho w_r)^2}{\rho^2} \end{aligned}$$

<u>`Vxy, Yyz, Vxz` — *Crossflow velocity magnitude*</u>

$$\begin{aligned} V_{xy} &= \sqrt{u^2 + v^2} \\ V_{yz} &= \sqrt{v^2 + w^2} \\ V_{xz} &= \sqrt{u^2 + w^2} \end{aligned}$$

<u>`M` — *Mach number*</u>

$$M = \frac{V}{a}$$

`a` — *Speed of sound*

$$\begin{aligned} a &= \sqrt{\gamma_\infty R_\infty T Z} \\ &= \sqrt{\gamma_\infty R_\infty T} \end{aligned}$$

Z contains the local real gas effects, R/R_∞.

`s` — *Entropy*

For frozen and finite-rate chemistry,

$$s - s_\infty = \sum_{i=1}^{ns} C_i \frac{R_u}{MW_i} (\mathtt{Sfun}_{i,T} - \mathtt{Sfun}_{i,T_\infty}) - R_{mix} \ln \left(\frac{p}{p_\infty} \right)$$

where `Sfun` is a function used to compute the entropy for a species as a function of temperature, from the curve fit data in the *.chm* file.

For a perfect gas,

$$s - s_\infty = (c_v)_\infty \ln \frac{p/p_\infty}{(\rho/\rho_\infty)^{\gamma_\infty}}$$

where $(c_v)_\infty = R_\infty/(\gamma_\infty - 1)$. Here s_∞ may be assumed to be zero.

`S` — *Strain rate magnitude*

$$S = \sqrt{2 S_{ij} S_{ij}}$$

`Sxx, Sxy, Sxz, Syx, Syy, Syz, Syz, Szx, Szy, Szz` — *Strain rate components*

$$S_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

`W` — *Rotation rate magnitude or vorticity magnitude*

$$\Omega = \sqrt{2 W_{ij} W_{ij}} = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}$$

The magnitude of the rotation rate tensor is equivalent to the vorticity magnitude.

`Wxx, Wxy, Wxz, Wyx, Wyy, Wyz, Wyz, Wzx, Wzy, Wzz` — *Rotation rate components*

$$W_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i} \right)$$

`wx, wy, wz` — *Vorticity components*

$$\begin{aligned} \omega_x &= \frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} \\ \omega_y &= \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \\ \omega_z &= \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \end{aligned}$$

`helicity` — *Helicity*

$$\text{helicity} = \mathbf{\Omega} \cdot \mathbf{V} = \omega_x u + \omega_y v + \omega_z w$$

`swirl` — *Swirl*

$$\text{swirl} = \frac{\mathbf{\Omega} \cdot \mathbf{V}}{\rho V^2} = \frac{\omega_x u + \omega_y v + \omega_z w}{\rho V^2}$$

`dila` — *Dilatation*

$$\text{dila} = \nabla \cdot \mathbf{V} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$$

`localpha` — *Local α* (o)

Let v_{ds} be the velocity in the "downstream" direction and v_{up} be the velocity in the "upward" direction as defined by the aerodynamic axes set in the grid file or specified with the `orientation` command. Local α is then defined as:

$$\alpha = \arctan\left(\frac{v_{up}}{v_{ds}}\right)$$

`locbeta` — *Local β* (o)

Let v_{ss} be the velocity in the "sidestream" direction as defined by the aerodynamic axes set in the grid file or specified with the `orientation` command, and let V be the magnitude of the velocity vector. Local β is then defined as:

$$\beta = \arcsin\left(\frac{v_{ss}}{V}\right)$$

`mu` — *Total viscosity coefficient*

$$\mu = \mu_l + \mu_t$$

`mul` — *Laminar viscosity coefficient*

For frozen and finite-rate chemistry,

$$\mu_l = \sum_{i=1}^{ns} \left(\frac{X_i \mu_i}{\sum_{i=1}^{ns} X_j \phi_{i,j}} \right)$$

where

$$\mu_i = \mu_0 \left(\frac{T}{T_\mu}\right)^{\frac{3}{2}} \left(\frac{T_\mu + S_\mu}{T + S_\mu}\right)$$

$$\phi_{i,j} = \frac{[1 + (\mu_i/\mu_j)^{1/2}(MW_j/MW_i)^{1/4}]^2}{(8 + 8MW_i/MW_j)^{1/2}}$$

The molecular weight of each species is obtained from the thermodynamic data in the chemistry (*.chm*) file, and the values of μ_0, T_μ, and S_μ for each species are obtained from the transport data in the *.chm* file.

For a perfect gas,

$$\mu_l = \mu_0 \left(\frac{T}{T_0}\right)^{\frac{3}{2}} \left(\frac{T_0 + S_1}{T + S_1}\right)$$

where (Pratt and Whitney, pp. 29,86–87):

$$\begin{aligned}
\mu_0 &= 0.37445 \times 10^{-6} \text{ slug/ft-sec} &&= 1.79287 \times 10^{-5} \text{ kg/m-sec} \\
T_0 &= 518.67\ ^\circ\text{R} &&= 288.15 \text{ K} \\
S_1 &= 216.0\ ^\circ\text{R} &&= 120 \text{ K}
\end{aligned}$$

`kappa` — *Thermal conductivity coefficient*

For frozen and finite-rate chemistry,

$$\kappa = \sum_{i=1}^{ns} \left(\frac{X_i \kappa_i}{\sum_{i=1}^{ns} X_j \phi_{i,j}}\right)$$

where

$$\begin{aligned}
\kappa_i &= \kappa_0 \left(\frac{T}{T_\kappa}\right)^{\frac{3}{2}} \left(\frac{T_\kappa + S_\kappa}{T + S_\kappa}\right) \\
\mu_i &= \mu_0 \left(\frac{T}{T_\mu}\right)^{\frac{3}{2}} \left(\frac{T_\mu + S_\mu}{T + S_\mu}\right) \\
\phi_{i,j} &= \frac{[1 + (\mu_i/\mu_j)^{1/2}(MW_j/MW_i)^{1/4}]^2}{(8 + 8MW_i/MW_j)^{1/2}}
\end{aligned}$$

The molecular weight of each species is obtained from the thermodynamic data in the chemistry (*.chm*) file, and the values of κ_0, T_κ, S_κ, μ_0, T_μ, and S_μ for each species are obtained from the transport data in the *.chm* file.

For a perfect gas,

$$\kappa = \mu_l \frac{c_p}{P_r}$$

where $c_p = R_\infty \gamma_\infty / (\gamma_\infty - 1)$.

`delta` — *Boundary layer thickness* (s)

The edge of the boundary layer is determined according to the criteria established by the `boundary layer edge` command.

`delta1 or delta*` — *Boundary layer displacement thickness* (s)

$$\delta_1 = \int_0^\delta \left(1 - \frac{\rho V}{\rho_\delta V_\delta}\right) ds = \int_0^\delta ds - \frac{1}{\rho_\delta V_\delta} \int_0^\delta \rho V \, ds$$

where V is the total velocity magnitude.

Starting at the wall and advancing one point at a time along the grid line intersecting the wall, the integrals in the second equation are accumulated. At each grid point, the values of the subscripted variables are assumed to be the value at the current grid point. The value of the equation is then computed and compared to the value at the previous iteration. If the change meets the criteria specified by the `boundary layer edge` command, the iteration stops and the thickness will be set to the accumulated arc length along the grid line (i.e., the value of the first integral). (Algorithm compliments of R. H. Bush.)

`delta2 or THETA` — *Boundary layer momentum thickness* (s)

$$\delta_2 = \int_0^\delta \frac{\rho V}{\rho_\delta V_\delta}\left(1 - \frac{V}{V_\delta}\right) ds = \frac{1}{\rho_\delta V_\delta}\int_0^\delta \rho V\, ds - \frac{1}{\rho_\delta V_\delta^2}\int_0^\delta \rho V^2\, ds$$

The method described for the boundary layer displacement thickness `delta1` is used to evaluate the integral.

`delta3` — *Boundary layer energy thickness* (s)

$$\delta_3 = \int_0^\delta \frac{\rho V}{\rho_\delta V_\delta}\left(1 - \frac{V^2}{V_\delta^2}\right) ds = \frac{1}{\rho_\delta V_\delta}\int_0^\delta \rho V\, ds - \frac{1}{\rho_\delta V_\delta^3}\int_0^\delta \rho V^3\, ds$$

The method described for the boundary layer displacement thickness `delta1` is used to evaluate the integral.

`deltau` — *Boundary layer velocity thickness* (s)

$$\delta_u = \int_0^\delta \left(1 - \frac{V}{V_\delta}\right) ds = \int_0^\delta ds - \frac{1}{V_\delta}\int_0^\delta V\, ds$$

The method described for the boundary layer displacement thickness `delta1` is used to evaluate the integral. Note that the velocity thickness is the same as the incompressible displacement thickness.

`THETAinc` — *Boundary layer incompressible momentum thickness* (s)

$$\theta_{\text{inc}} = \int_0^\delta \frac{V}{V_\delta}\left(1 - \frac{V}{V_\delta}\right) ds = \frac{1}{V_\delta}\int_0^\delta V\, ds - \frac{1}{V_\delta^2}\int_0^\delta V^2\, ds$$

The method described for the boundary layer displacement thickness `delta1` is used to evaluate the integral. The incompressible momentum thickness is often used to compute an incompressible shape factor for evaluating flow control devices.

`Q` — *Heat transfer rate* (s)

$$Q = \kappa\frac{\partial T}{\partial n} = \kappa\nabla T\cdot\mathbf{n} = \kappa\left(\frac{\partial T}{\partial x}\frac{\partial x}{\partial n} + \frac{\partial T}{\partial y}\frac{\partial y}{\partial n} + \frac{\partial T}{\partial z}\frac{\partial z}{\partial n}\right)$$

The equation is evaluated using the same process described in the derivation of `tau`.

`tau, taux, tauy, tauz` — *Total (laminar+turbulent) shear stress* (s)

$$\boldsymbol{\tau} = \mu\frac{\partial \mathbf{V}_{tan}}{\partial n}$$

where μ is the total (laminar+turbulent) viscosity

$$\mu = \mu_l + \mu_t$$

and $\mathbf{V}_{tan}$ is the tangential velocity vector determined as follows:

$$\mathbf{V}_{tan} = \mathbf{V} - (\mathbf{V}\cdot\mathbf{n})\mathbf{n}$$

The magnitude of the shear stress is computed as

$$\tau = \mu\sqrt{\left(\frac{\partial u_{tan}}{\partial n}\right)^2 + \left(\frac{\partial v_{tan}}{\partial n}\right)^2 + \left(\frac{\partial w_{tan}}{\partial n}\right)^2}$$

where the partial derivatives are evaluated using the chain rule:

$$\frac{\partial f}{\partial n} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial n} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial n} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial n}$$

The partial derivatives are computed using the equations presented in C.7.

The shear stress acts in the direction of the velocity, so the components are:

$$\tau_x = \tau\frac{u_t}{V_t}$$
$$\tau_y = \tau\frac{v_t}{V_t}$$
$$\tau_z = \tau\frac{w_t}{V_t}$$

If wall function boundary conditions were used in the flow solver, CFPOST will also use wall functions when computing the shear stress.

`Cf, Cfx, Cfy, Cfz` — *Skin friction coefficient* (s)

The total skin friction coefficient is the shear stress magnitude normalized by the freestream dynamic pressure.

$$C_f = \frac{\tau}{\rho_\infty V_\infty^2/2}$$

Similarly, the Cartesian components of the skin friction coefficient are defined as

$$(C_f)_x = \frac{\tau_x}{\rho_\infty V_\infty^2/2}$$
$$(C_f)_y = \frac{\tau_y}{\rho_\infty V_\infty^2/2}$$
$$(C_f)_z = \frac{\tau_z}{\rho_\infty V_\infty^2/2}$$

Note that the freestream values used are the reference values specified with the Wind-US `FREESTREAM` keyword and may differ substantially from those at the local boundary layer edge.

`uu, uv, uw, vu, vv, vw, wu, wv, ww` — *Turbulent stress components* (s)

$$u_i''u_j'' = \overline{\rho u_i''u_j''}/\bar{\rho} = -\left[2\mu_t\left(S_{ij} - S_{kk}\delta_{ij}/3\right) - 2/3\rho k\delta_{ij}\right]$$

Turbulent stresses can be computed for any of the structured grid turbulence models. The formula above represents the Boussinesq approximation used by most models. For the Spalart-Allmaras turbulence model, the turbulent kinetic energy is estimated by means of a generalized form of Bradshaw's relation:

$$\rho k = \mu_t\sqrt{2S_{ij}S_{ij}}/0.31$$

Algebraic Reynolds stress models use a more complex form for the stresses that involves non-linear expressions of the strain and rotation rate tensors.

`y+` — *Non-dimensional boundary layer wall coordinate* (s)

$$y^+ = \frac{y}{\nu_{wall}} u_\tau$$

where

$$u_\tau = \sqrt{\frac{\tau_{wall}}{\rho_{wall}}}$$
$$\nu_{wall} = \frac{\mu_{wall}}{\rho_{wall}}$$

and here y is the distance from the wall, measured along the grid line intersecting the wall.

`u+` — *Non-dimensional boundary layer velocity* (s)

$$u^+ = \frac{V}{u_\tau}$$

where $u_\tau = \sqrt{\tau_{wall}/\rho_{wall}}$.

`k+` — *Non-dimensional turbulent kinetic energy* (s)

$$k^+ = k/u_\tau^2$$

where $u_\tau = \sqrt{\tau_{wall}/\rho_{wall}}$.

`epsilon+` — *Non-dimensional turbulent dissipation rate* (s)

$$\epsilon^+ = \epsilon\mu_l/\left(\rho u_\tau^4\right)$$

where $u_\tau = \sqrt{\tau_{wall}/\rho_{wall}}$.

`uu+, uv+, uw+, vu+, vv+, vw+, wu+, wv+, ww+` — *Non-dimensional turbulent stress components* (s)

$$\left(u_i'' u_j''\right)^+ = \overline{\rho u_i'' u_j''}/\left(\rho u_\tau^2\right)$$

where $u_\tau = \sqrt{\tau_{wall}/\rho_{wall}}$.

`Mt` — *Turbulent Mach number*

$$Mt = \sqrt{2k}/a$$

`Rt` — *Turbulent Reynolds number* (Ladd and Kral, 1992, p. 3)

$$Rt = \frac{\rho k^2}{\mu_l \epsilon}$$

`Ry` — *Reynolds number based on y* (s) (Ladd and Kral, 1992, p. 3)

$$Ry = y\frac{\rho}{\mu_l}\sqrt{k}$$

where here y is the distance from the wall, measured along the grid line intersecting the wall.

`fmuJL, fmuCH, fmuSP, fmuLB` — *Turbulence model damping functions* (s) (Ladd and Kral, 1992, p. 3)

$$
\begin{aligned}
(f_\mu)_{JL} &= e^{-2.5/(1+Rt/50)} \\
(f_\mu)_{CH} &= 1 - e^{-0.0115y^+} \\
(f_\mu)_{SP} &= \left(1 + \frac{3.45}{\sqrt{Rt}}\right) \tanh \frac{y^+}{70} \\
(f_\mu)_{LB} &= \left(1 + \frac{20.5}{Rt}\right) \left(1 - e^{-0.0165Ry}\right)^2
\end{aligned}
$$

`C-`*`species`* — *Mass fraction of species*

Using `O2` as an example for *`species`*,

$$\texttt{C-O2} = \frac{\rho_{\mathrm{O2}}}{\rho}$$

`X-`*`species`* — *Mole fraction of species*

Using `O2` as an example for *`species`*,

$$\texttt{X-O2} = \frac{\texttt{C-O2}}{MW_{\mathrm{O2}}} \sum_{i=1}^{ns} MW_i$$

`Veff` — *Effective collision frequency*

$$V_{\mathit{eff}} = C \frac{p}{T} \sqrt{T_e}$$

where C is a function of the gas, p is the pressure in torr (i.e., millimeters of mercury), T is the temperature in K, and T_e is the electron temperature. In CFPOST, C is assumed to be 7.3×10^9, the value for air, and T_e is assumed equal to T.

`MW` — *Molecular weight*

For frozen and finite-rate chemistry,

$$MW = \frac{1}{\sum_{i=1}^{ns} \frac{C_i}{MW_i}}$$

where C_i is the mass fraction of species i. The molecular weights for all the species may be listed using the `show chemistry` command.

For a perfect gas,

$$MW = \frac{R_u}{R_\infty}$$

`R` — *Gas constant*

For frozen and finite-rate chemistry,

$$R = \frac{R_u}{MW}$$

For a perfect gas,

$$R = R_\infty$$

<u>`cp` — *Specific heat at constant pressure*</u>

For frozen and finite-rate chemistry,

$$c_p = \sum_{i=1}^{ns} C_i \frac{R_u}{MW_i} \texttt{Cpfun}_{i,T}$$

where `Cpfun` is a function used to compute the specific heat at constant pressure for a species as a function of temperature, from the curve fit data in the *.chm* file.

For a perfect gas,

$$c_p = \frac{\gamma_\infty}{\gamma_\infty - 1} R_\infty$$

<u>`cv` — *Specific heat at constant volume*</u>

$$c_v = c_p - R$$

<u>`gamma` — *Ratio of specific heats*</u>

For frozen and finite-rate chemistry,

$$\gamma = \frac{c_p}{c_v}$$

For a perfect gas,

$$\gamma = \gamma_\infty$$

<u>`hf` — *Heat of Formation*</u>

For frozen and finite-rate chemistry,

$$h_f = \sum_{i=1}^{ns} C_i [h_{f(298)}]_i$$

For a perfect gas,

$$h_f = 0$$

C.2 Thermodynamic Functions

The functions `Cpfun`, `Hfun`, and `Sfun` appearing in some of the above equations for frozen and finite-rate chemistry flows are used to compute thermodynamic properties for a species as a function of temperature. Properties are computed using curve fit equations, with the coefficients supplied in a chemistry data (*.chm*) file. The *.chm* file to be used must be specified using the CFPOST `chemistry` command.

Three different (but related) formats are available for the type of curve fits being specified in the *.chm* file — `SPARKCRV`, `WINDNASA`, and `NASA3287`.

`Cpfun` — Molar specific heat at constant pressure

Function `Cpfun` computes the specific heat per mole at constant pressure for a species. For the `SPARKCRV` and `WINDNASA` formats, the curve fit formula is

$$\frac{C_p}{R_u} = a_1 + a_2T + a_3T^2 + a_4T^3 + a_5T^4$$

For the NASA3287 format, the formula is

$$\frac{C_p}{R_u} = a_1T^{-2} + a_2T^{-1} + a_3 + a_4T + a_5T^2 + a_6T^3 + a_7T^4 + a_8T^5$$

Note that the value returned by `Cpfun` is dimensionless.

`Hfun` — Molar absolute enthalpy

Function `Hfun` computes the enthalpy per mole for a species. For the `SPARKCRV` and `WINDNASA` formats, the curve fit formula is

$$\frac{H}{R_u} = a_1T + a_2\frac{T^2}{2} + a_3\frac{T^3}{3} + a_4\frac{T^4}{4} + a_5\frac{T^5}{5} + b_1$$

For the NASA3287 format, the formula is

$$\frac{H}{R_u} = -a_1T^{-1} + a_2\ln T + a_3T + a_4\frac{T^2}{2} + a_5\frac{T^3}{3} + a_6\frac{T^4}{4} + a_7\frac{T^5}{5} + a_8\frac{T^6}{6} + b_1$$

For the `SPARKCRV` curve type, the enthalpy reference state is at $T = 0$ K. I.e., the curve fit for H/R_u actually computes

$$\frac{H}{R_u} = \frac{H_{f(0)} + \Delta H_{(0\to T)}}{R_u}$$

where $H_{f(0)}$ is the heat of formation at 0 K, and $\Delta H_{(0\to T)}$ is the change in enthalpy between 0 K and T.

However, for the `WINDNASA` and `NASA3287` formats the enthalpy reference state is at $T = 298.15$ K. I.e., for these formats the curve fit computes

$$\frac{H'}{R_u} = \frac{H_{f(298)} + \Delta H_{(298\to T)}}{R_u}$$

where $H_{f(298)}$ is the heat of formation at 298.15 K, and $\Delta H_{(298\to T)}$ is the change in enthalpy between $298.15 K and T$.

Thus, for the `WINDNASA` and `NASA3287` formats, function `Hfun` adjusts the value to shift the reference state to 0 K. I.e., the actual returned value is

$$\begin{aligned}\frac{H}{R_u} &= \frac{H'}{R_u} + \frac{H_{(shift)}}{R_u}\\ &= \frac{H_{f(298)} + \Delta H_{(298\to T)}}{R_u} + \frac{H_{f(0)} - H_{f(298)} + \Delta H_{(0\to 298)}}{R_u}\\ &= \frac{H_{f(0)} + \Delta H_{(0\to T)}}{R_u}\end{aligned}$$

Note that the value returned by `Hfun` has units of K.

`Sfun` — Molar entropy

Function `Sfun` computes the entropy per mole for a species. For the `SPARKCRV` and `WINDNASA` formats, the curve fit formula is

$$\frac{S}{R_u} = a_1 \ln T + a_2 T + a_3 \frac{T^2}{2} + a_4 \frac{T^3}{3} + a_5 \frac{T^4}{4} + b_2$$

For the NASA3287 format, the formula is

$$\frac{S}{R_u} = -a_1 \frac{T^{-2}}{2} - a_2 T^{-1} + a_3 \ln T + a_4 T + a_5 \frac{T^2}{2} + a_6 \frac{T^3}{3} + a_7 \frac{T^4}{4} + a_8 \frac{T^5}{5} + b_2$$

It should be noted that the entropy is defined by the equation

$$dS = C_p \frac{dT}{T} - R_u \frac{dp}{p}$$

Thus

$$\begin{aligned} \frac{S}{R_u} &= \int \frac{1}{R_u} dS \\ &= \int \frac{C_p}{R_u} \frac{dT}{T} - \int \frac{dp}{p} \end{aligned}$$

and `Sfun` computes the temperature portion of the above equation, $\int (C_p/R_u) dT/T$.

Note also that the value returned by `Sfun` is dimensionless.

C.3 Integration Formulas

The area to be integrated is considered to be a collection of individual polygons. A surface may be defined directly by a `subset` or `surface` command. If the dimensions of the surface is i points by j points, there will be $i - 1$ times $j - 1$ polygons that define the surface. The perimeter of the surface is described by the line segments that join adjacent grid points at the extreme bounds of the subset. A surface may also be defined as the intersection of a cutting plane defined by a `cut` command with a computational volume defined by a `subset` command. The polygons created by passing the cutting plane through the cells that make up the volume will be those processed by the integration.

For surfaces defined directly, each polygon will have exactly four verticies defined by the grid points that make up the polygon. The area will be determined by computing the magnitude of the cross products of the diagonals. For surfaces defined by a cutting plane, each polygon will have from three to six verticies defined by the intersection of the cutting plane with the cell edges. The area will be determined by decomposing the polygon into triangles and adding their individual areas. The unit normal vector for each polygon will be defined to be the unit vector in the direction of the vector sum of the normal vectors at the vertices.

For any cell, the value of a particular property for that cell (pressure, temperature, density, etc.) will be defined to be the average of the values at the verticies for that cell; i.e., for a cell with N_v vertices,

$$f_{cell} = \frac{1}{N_v} \sum_{i=1}^{N_v} f_i$$

C.4 Area and Mass Average Integration

Averages over a group of N_c cells are computed as shown below. In these formulas the variables being summed are cell averages as defined above.

Area Average

$$f_{area} = \frac{\sum_{i=1}^{N_c} f_i A_i}{\sum_{i=1}^{N_c} A_i}$$

Mass Average

$$f_{mass} = \frac{\sum_{i=1}^{N_c} \rho_i f_i A_i}{\sum_{i=1}^{N_c} \rho_i A_i}$$

Mass Flux Average

$$f_{massflux} = \frac{\sum_{i=1}^{N_c} \rho_i f_i (\mathbf{V}_i \cdot \mathbf{A}_i)}{\sum_{i=1}^{N_c} \rho_i (\mathbf{V}_i \cdot \mathbf{A}_i)}$$

Arithmetic Mean

$$f = \frac{1}{N_c} \sum_{i=1}^{N_c} f_i$$

$$\begin{aligned}\sigma &= \sqrt{\frac{1}{N_c - 1} \sum_{i=1}^{N_c} (f_i - f)^2} \\ &= \sqrt{\frac{1}{N_c - 1} \left[\left(\sum_{i=1}^{N_c} f_i^2 \right) - N_c f^2 \right]}\end{aligned}$$

C.5 Flux Integration

Mass Flux

$$\dot{m} = \rho (\mathbf{V} \cdot \mathbf{n}) A$$

Momentum Flux

$$\mathbf{F}_{momentum} = \rho \mathbf{V} (\mathbf{V} \cdot \mathbf{n}) A$$

Pressure Flux

$$\mathbf{F}_{pressure} = p A \mathbf{n}$$

Total Thrust

$$\mathbf{F}_{total} = \mathbf{F}_{momentum} + \mathbf{F}_{pressure}$$

Gross Thrust

$$\mathbf{F}_G = (p - p_\infty)A\mathbf{n}$$

C.6 Force and Moment Integration (not yet done!)

$$\mathbf{F} = \mathbf{F}_{pressure} + \mathbf{F}_{viscous}$$

C.7 Derivatives of Variables With Respect to Cartesian Coordinates

Computing partial derivatives of variables with respect to Cartesian coordinates requires the calculation of the metric coefficients. Define the Jacobian as follows:

$$J = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \zeta} \end{vmatrix}$$

The metric coefficients are then computed as follows:

$$\begin{aligned}
\frac{\partial \xi}{\partial x} &= J^{-1}\left(\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \zeta} - \frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \eta}\right) & \frac{\partial \xi}{\partial y} &= J^{-1}\left(\frac{\partial x}{\partial \zeta}\frac{\partial z}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \zeta}\right) & \frac{\partial \xi}{\partial z} &= J^{-1}\left(\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \eta}\right) \\
\frac{\partial \eta}{\partial x} &= J^{-1}\left(\frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \zeta}\right) & \frac{\partial \eta}{\partial y} &= J^{-1}\left(\frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial z}{\partial \xi}\right) & \frac{\partial \eta}{\partial z} &= J^{-1}\left(\frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \xi} - \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \zeta}\right) \\
\frac{\partial \zeta}{\partial x} &= J^{-1}\left(\frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \xi}\right) & \frac{\partial \zeta}{\partial y} &= J^{-1}\left(\frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \xi} - \frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \eta}\right) & \frac{\partial \zeta}{\partial z} &= J^{-1}\left(\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}\right)
\end{aligned}$$

Given the above, it is then possible to compute the partial derivative of a variable with respect to a Cartesian coordinate. Using the chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial f}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial f}{\partial \zeta}\frac{\partial \zeta}{\partial x}$$

It is often necessary to know the vector normal to computational surface. First we define vectors tangent to a line in which only one of the computational coordinates vary:

$$\begin{aligned}
\mathbf{V}_\xi &= \frac{\partial x}{\partial \xi}\mathbf{i} + \frac{\partial y}{\partial \xi}\mathbf{j} + \frac{\partial z}{\partial \xi}\mathbf{k} \\
\mathbf{V}_\eta &= \frac{\partial x}{\partial \eta}\mathbf{i} + \frac{\partial y}{\partial \eta}\mathbf{j} + \frac{\partial z}{\partial \eta}\mathbf{k} \\
\mathbf{V}_\zeta &= \frac{\partial x}{\partial \zeta}\mathbf{i} + \frac{\partial y}{\partial \zeta}\mathbf{j} + \frac{\partial z}{\partial \zeta}\mathbf{k}
\end{aligned}$$

The vector normal to a constant computational surface is then the cross product of the two vectors whose computational coordinates vary on the surface:

$$\mathbf{N}_\xi = \mathbf{V}_\eta \times \mathbf{V}_\zeta$$
$$\mathbf{N}_\eta = \mathbf{V}_\zeta \times \mathbf{V}_\xi$$
$$\mathbf{N}_\zeta = \mathbf{V}_\xi \times \mathbf{V}_\eta$$

Given a normal vector, one can then simply form a unit normal vector and directional derivatives.

$$\mathbf{n} = \frac{\mathbf{N}}{|\mathbf{N}|} = n_x\mathbf{i} + n_y\mathbf{j} + n_z\mathbf{k}$$

where

$$\frac{\partial x}{\partial n} = n_x \qquad \frac{\partial y}{\partial n} = n_y \qquad \frac{\partial z}{\partial n} = n_z$$

C.8 Derivatives of Variables With Respect to the Computational Coordinates

Derivatives of values with respect to computational coordinates may be either first or second order accurate depending on the need. Where normal vectors are not of a concern, second order central differences are used except where a boundary is involved and then first order forward or backward differences are used:

$$\frac{\partial f}{\partial \xi} \approx \delta f = \frac{f_{i+1} - f_{i-1}}{2} \qquad \text{Central difference} \qquad (1)$$
$$\frac{\partial f}{\partial \xi} \approx \Delta f = f_{i+1} - f_i \qquad \text{Forward difference} \qquad (2)$$
$$\frac{\partial f}{\partial \xi} \approx \nabla f = f_i - f_{i-1} \qquad \text{Backward difference} \qquad (3)$$

For normal derivatives, first or second order may be selected by the `first order` or `second order` command or parameter on the `integrate force` command. Because of the internal structure of CFPOST, these derivatives are always forward derivatives:

$$\frac{\partial f}{\partial \xi} \approx f_{i+1} - f_i \qquad \text{First order} \qquad (4)$$
$$\frac{\partial f}{\partial \xi} \approx \frac{-f_{i+2} + 4f_{i+1} - 3f_i}{4} \qquad \text{Second order} \qquad (5)$$

C.9 Engine Analysis Calculations

C.9.1 Distortion — SAE Definition (Society of Automotive Engineers, 1978)

A low-pressure region is defined as a range of rake probes on a particular ring whose total pressures fall below the average total pressure for that ring.

Circumferential Intensity

$$\mathit{Intensity} = \left(\frac{\Delta PC}{P}\right)_{\mathit{Ring\ i}} = \frac{\mathit{PAV}_{\mathit{Ring\ i}} - \mathit{PAVLOW}_{\mathit{Ring\ i}}}{\mathit{PAV}_{\mathit{Ring\ i}}}$$

where

$$\mathit{PAV}_{\mathit{Ring\ i}} = \text{Ring-averaged total pressure}$$
$$\mathit{PAVLOW}_{\mathit{Ring\ i}} = \text{Ring-averaged total pressure over all low-pressure regions}$$

Circumferential Extent

$$\mathit{Extent} = \text{Cumulative angular extent of all low-pressure regions, in degrees}$$

Multiple-per-revolution (MPR)

$$\mathit{MPR}_{\mathit{Ring\ i}} = \frac{\sum_{k=1}^{Q}\left[\left(\frac{\Delta PC}{P}\right)_i k\theta_i k^-\right]}{\max\left[\left(\frac{\Delta PC}{P}\right)_i k\theta_i k^-\right]}$$

where

$$Q = \text{Number of low-pressure regions}$$
$$\theta_i k^- = \text{Extent of low-pressure region } k \text{ on the } i\text{'th ring}$$

Radial Intensity

$$\mathit{Intensity} = \left(\frac{\Delta PR}{P}\right)_{\mathit{Ring\ i}} = \frac{\mathit{PFAV} - \mathit{PAV}_{\mathit{Ring\ i}}}{\mathit{PFAV}}$$

where

$$\mathit{PFAV} = \text{Area-weighted face-average total pressure}$$

C.9.2 Distortion — GE Definition

Circumferential Intensity

$$\mathit{Intensity} = \frac{\mathit{PAV}_{\mathit{Ring\ i}} - \mathit{PMIN}_{\mathit{Ring\ i}}}{\mathit{PAV}}$$

where

$$\mathit{PAV}_{\mathit{Ring\ i}} = \text{Area-averaged total pressure of ring } i$$
$$\mathit{PMIN}_{\mathit{Ring\ i}} = \text{Minimum total pressure of ring } i$$
$$\mathit{PAV} = \text{Area-averaged total pressure over the complete face}$$

Circumferential Extent

The circumferential extent is the same as the SAE definition.

Radial Intensity

The radial intensity is the same as the SAE definition.

Shape Factor

$$\mathit{Shape\ Factor} = \frac{\mathit{GE\ ring\ circumferential\ intensity}}{\mathit{SAE\ ring\ circumferential\ intensity}}$$

References

Ladd, J. A., and Kral, L. D. (1992) “Development and Application of a Zonal k-ϵ Turbulence Model for Complex 3-D Flowfields,” AIAA/SAE/ASME/ASEE 28th. Joint Propulsion Conference and Exhibit, AIAA Paper 92-3176.

Pratt and Whitney Aircraft Company “Aeronautical Vestpocket Handbook,” 19th. ed., Part Number 79500.

Society of Automotive Engineers (1978) “Gas Turbine Engine Inlet Flow Distortion Guidelines,” SAE-ARP-1420.